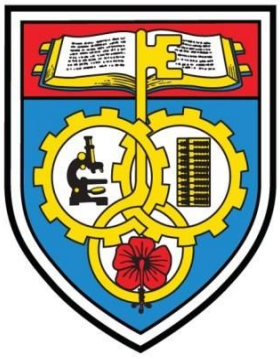
Fine-Grained Clothing Image Retrieval with Faster R-CNN

By

Ong Tun Ying



TARC

TUNKU ABDUL RAHMAN
UNIVERSITY COLLEGE

FACULTY OF COMPUTING AND
INFORMATION TECHNOLOGY

TUNKU ABDUL RAHMAN UNIVERSITY COLLEGE
KUALA LUMPUR

ACADEMIC YEAR
2019/20

Fine-Grained Clothing Image Retrieval with Faster R-CNN

By

Ong Tun Ying

Supervisor: Dr. Lim Khai Yin

A project report submitted to the
Faculty of Computing and Information Technology
in partial fulfillment of the requirement for the
Bachelor of Computer Science (Honours)

Department of Computer Science and Embedded Systems
Faculty of Computing and Information Technology
Tunku Abdul Rahman University College
Kuala Lumpur

2019/20

Declaration

The project submitted herewith is a result of my own efforts in totality and in every aspect of the project works. All information that has been obtained from other sources had been fully acknowledged. I understand that any plagiarism, cheating or collusion or any sorts constitutes a breach of TAR University College rules and regulations and would be subjected to disciplinary actions.

_____

Ong Tun Ying

Bachelor of Computer Science (Honours) in Software Engineering

ID: 18WMR08499

Abstract

It is clear that image retrieval based application is still considerably niche in the fashion and clothing industry. Many have attempted to develop frameworks and methods to build such applications, while they succeed to make image retrieval out of fashion images, the lack of performance and accuracy causes them to be unsuitable for real-world usage. This research is another such attempt to solve the existing problems of lack of performance and accuracy. This research covers topics such as pattern recognition, convolutional neural networks (CNN). CNN architectures, and methodologies to solve the underlying problems such as choosing the correct dataset, dataset conversion, object detection, and transfer learning.

Acknowledgement

My utmost gratitude is given to my research supervisor, Dr. Lim Khai Yin for her dedication and her effort to get involved in assisting in this project throughout the process until completion. The successful completion of the project would not be possible without her.

I also owe my gratitude and thanks to my classmates, whom have been supportive and kind to me when I need their assistance and advice. Overcoming the challenges and hardships for this project required more than just academic support. That being said, I have countless people to thank for listening and motivating me relentlessly over the past three years. I cannot begin to express my gratitude and appreciation for their friendship and support.

Last but not least, without my dearest family, none of these would be possible. To my parents, thank you for always being there for all my ups and downs for the past three years. Thank you for always providing me with a safe and peaceful home that I could return to after each exhausting semester. I am truly grateful for the unconditional love given to me and I would never take it for granted.

Table of Contents

Declaration 3

Abstract 4

Acknowledgement 5

1 Introduction 9

1.1 Problem Statement 9

1.2 Research Objectives 9

1.3 Research Scope 10

2 Literature Review 12

2.1 Pattern Recognition 12

2.1.1 Patterns and Features 13

2.1.2 Statistical Approach to Pattern Recognition 13

2.1.3 Structural Approach to Pattern Recognition 13

2.2 Neural Networks 14

2.2.1 Neural Network Architectures 14

2.2.2 Training Process 16

2.3 Convolutional Neural Network (CNN) 17

2.3.1 Pooling Layers 18

2.3.2 Process 18

2.3.3 Backpropagation 20

2.4 CNN Architectures 20

2.4.1 AlexNet 20

2.4.2 VGGNet 22

2.4.3 ResNet 24

2.5 Object Detection CNN Architectures 26

2.5.1 R-CNN 27

2.5.2 Fast R-CNN 28

2.5.3 YOLO - You Only Look Once 28

2.6 Conclusion 29

3 Research Methodology 31

3.1 Proposed Framework 31

3.1.1 Dataset Conversion 32

3.1.2 Faster R-CNN on TensorFlow 33

3.1.3 Considerations Against Mask R-CNN 34

3.1.4 Transfer Learning on FasterRCNN-Inception-v2 using TensorFlow 34

3.1.5 Inferencing and Testing 34

3.2 Dataset 36

3.2.1 DeepFashion 36

3.2.2 Considerations Against DeepFashion2 37

3.3 Evaluation Method 38

3.3.1 Holdout 38

3.3.2 Evaluation of Classification 39
3.4 Summary 40

4 Theory Background 42
4.1 Inception-v1 42
4.2 Inception-v2 45
4.3 Faster R-CNN 48
4.4 Transfer Learning 49
4.4 Summary 51

5 Implementation and Testing 53
5.1 Sub-section 1 Heading 53
5.1.1 Sub-subsection Heading 53
5.2 Sub-section 2 Heading 53
5.3 Sub-section 1 Heading 53
5.3.1 Sub-subsection Heading 53
5.4 Sub-section 1 Heading 54
5.4.1 Sub-subsection Heading 54
5.5 Chapter Summary and Evaluation 54

6 System Deployment 56
6.1 System Backup and Risk Management 56
6.2 On-site Setup 56
6.3 Training Procedure 56
6.4 Follow-up 56
6.5 Chapter Summary and Evaluation 56

7 Discussions and Conclusion 58
7.1 Summary 58
7.2 Achievements 58
7.3 Contributions 58
7.4 Limitations and Future Improvements 58
7.5 Issues and Solutions 58

References 60

Chapter 1

Introduction

1 Introduction

In 2019, fashion and clothing image retrieval remains to be an advanced image processing framework which has high potential in serving needs such as computer aided fashion design (fashion recommendation system), customer profile analysis, and context-aided people identification. However, it is rare to find a framework which can recognize fashion categories and attributes through real-time surveillance cameras due the needs to track the human subjects, segmentation of clothing, and creating a library of categories and attributes for the clothings (Yang and Yu, 2011). Using DeepFashion's dataset that comes with 50 established categories and 1000 attributes to be used (Liu et al., 2016), along with TensorFlow's (Abadi et al., 2016) built-in object detection and tracking models like Faster R-CNN (Szegedy et al., 2015), a deep learning framework is built for fine-grained clothing image retrieval. This framework could potentially benefit fashion companies and ecommerce companies as well as any fashion enthusiast worldwide in acquiring data of fashion trends (Wired, 2019), to answer simple questions like: "What are the most favored colors for women's fashion in Malaysia?" without even asking anyone (Innovation Enterprise, n.d.).

1.1 Problem Statement

Though many researches have been performed to harvest the potential of fashion and clothing recognition, most researches focus on the reliability and performance of the deep learning models (Yang and Yu, 2011). Notwithstanding the importance of such, very few developments have been done in the fashion field to implement and to apply the deep learning models. However, the primary reasons for this lack of development are not due to the lack of interest, but rather due to many other sub-problems that are challenging to overcome in building such a system.

First, such a framework might involve sub-problems such as face detection and/or body tracking, human figure (poselets or body parts) or clothing (landmarks) segmentation, and effective clothing representations.

Second, the discerning of various subtle differences among fashion categories and attributes requires considerable computations to be considered well-achieved.

1.2 Research Objectives

- To explore every important topic that is needed to build a fine-grained clothing image retrieval framework.
- To analyze the topics, and choose the most preferred methods to build to framework. The framework needs to be able to solve problems mentioned in Section 1.1, which are:

  - ➢ To be able to detect/track objects based on an input, which can be still images or motion images.
  - ➢ To be able to retrieve clothing information based on fashion categories and attributes from the images.
- Design and build the framework.

1.3 Research Scope

In making sure that the proposed framework is able to solve real-world problems, as well as being relevant in the real-world, a scope is defined for this research, the topics which have been explored are:

- Theories related to pattern recognition.
- The artificial neural network.
- CNNs and publicised architectures based on it.
- Every possible methodologies and considerations in building the proposed framework, which includes:
  - ➢ The dataset that is needed.
  - ➢ The CNN architecture that is being referred to.
  - ➢ The existing frameworks/tools needed to build the framework.
  - ➢ Methods to evaluate the results.

Chapter 2

Research Background

2 Literature Review

In this chapter, relevant researches related to the project will be presented. Topics such as pattern recognition, neural networks, convolutional neural networks (CNN), and CNN architectures shall be discussed in detail in this chapter.

2.1 Pattern Recognition

The discipline of pattern recognition is concerned with the automatic discovery of fundamental structure in the input objects based on preexisting knowledge or statistical information extracted from the structure through the use of machine intelligence (Chakrabarty, 2010) (Howard, 2007).

In the past, two prominent approaches to pattern recognition are statistical, and structural. Today, artificial neural networks has provided an alternative - neural pattern recognition.

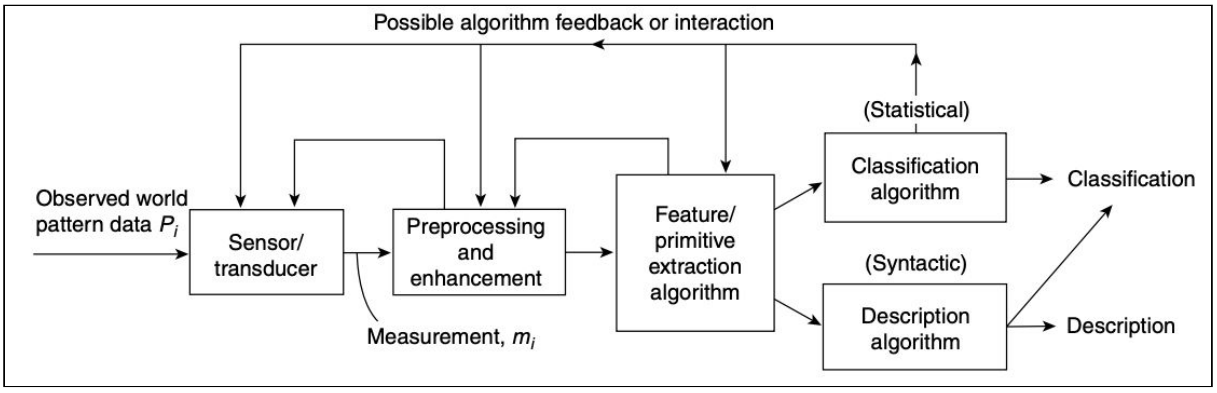


Figure 2.1 illustrates a generic pattern recognition system structure that is applicable to all three of the aforementioned approaches. Notice that it consists of a sensor(s), which in some cases, pre-classified data may also be used and are usually called the 'training data' (Schalkoff, 2007).

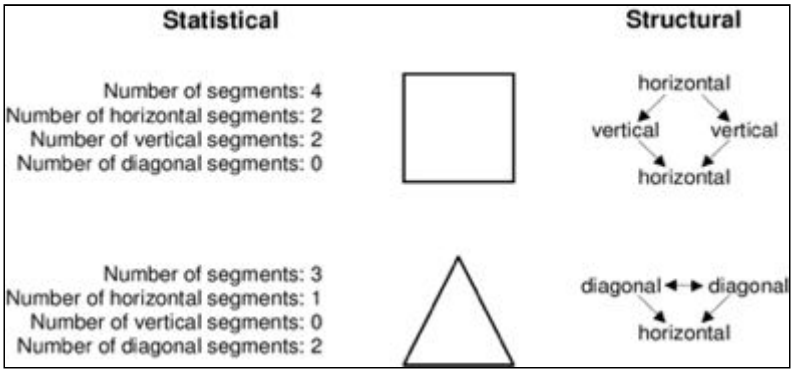


Figure 2.2: The statistical and structural approaches to pattern recognition applied to a common identification problem. The goal is to discriminate between the square and the triangle (Olszewski R. T. 2001). .

2.1.1 Patterns and Features

Patterns can be a set of measurements or observations, usually represented in vector notation. Whereas features are the measurements extracted from the patterns. Features may be symbolic, numeric, or both. Color is an example of a symbolic feature, whereas weight is an example of a numerical feature. Feature extraction algorithms are used to extract features from patterns. Extracted features may still contain 'noise' or unwanted measurements.

2.1.2 Statistical Approach to Pattern Recognition

The statistical approach defines that the measurements taken from N features are represented in N-dimensional pattern space as one point. The portion of the pattern space defines the principle of the classification into subspaces, each corresponding to a specific pattern class.Sometimes, the rejection of the classification of some patterns can lead to desirable outcomes.

Parametric classification methods, discriminant functions, clustering analysis, and fuzzy set reasoning are some techniques that use the statistical approach. When there is no prior knowledge of the information, the use of fuzzy set reasoning creates an alternative to the probabilistic approach (Zadeh, 1965).

However, the techniques above cause optimization problems. Besides, the quantification of the contribution of a particular feature towards the accuracy of classification also creates ambiguity. This together with the indeterminacy in the statically inter-relationship between features, causes their specification to become an exercise in educated guessing (Nandhakumar and Aggarwal, 1985). In general, all pattern features cannot be represented exactly in mathematical expressions and the evaluation of the effectiveness of feature ordering largely depends on human subjective decisions.

2.1.3 Structural Approach to Pattern Recognition

The structural approach defines that the idea based on the recursive description of complex patterns into simpler patterns just as sentences are built up by concatenating words, and words are built up by concatenating characters (Kepka, 1994). This results in both the classification and the structural description of an analyzed pattern by means of a set of pattern primitives (simplest sub-patterns) and their relationships.

Classification is performed via syntax analysis which determines if the pattern representation is correct in accordance with the given grammar. In the former case a complete syntactic description - a parsing tree - of the analyzed pattern is produced. Given the pattern cannot be successfully parsed in any possible classes of patterns then it is rejected. The parsing tree is constructed either from its root towards the bottom or from the bottom towards the top. In order to obtain a grammar representing the

structural information on the patterns, grammatical inference machine is required to conclude from a given asset of training patterns.

It should come to be aware that the costs between the complexity of the recognizer and the descriptive power of the language must be chosen. When it comes to multidimensional patterns – 2D and 3D images, high-dimensional pattern grammars are used such as tree grammars, plex grammars, web grammar, etc.

The limitations to this approach comes when the pure structural approach is not capable in handling numerical semantic information for recognition. Plus, when a pattern can be generated by more than one pattern grammar, ambiguity becomes an issue.

2.2 Neural Networks

Neural networks are a set of algorithms, modeled loosely after the human brain, that are designed to recognize patterns. They interpret sensory data through a kind of machine perception, labeling or clustering raw input. The patterns they recognize are numerical, contained in vectors, into which all real-world data, be it images, sound, text or time series, must be translated (Nicholson, n.d.).

A standard neural network (NN) consists of many connected processors called neurons, each producing a sequence of real-valued activations. Input neurons get activated through sensors perceiving the environment, other neurons get activated through weighted connections from previously active neurons (Schmidhuber, 2015).

The first breakthrough in Artificial Neural Network (ANN) was in the 1940s due to McCulloch and Pitts' pioneering work (McCulloch and Pitts, 1990). The second occurred in the 1960s with Rosenblatt's perceptron convergence theorem and Minsky and Papert's work showing the limitations of a simple perceptron (Nievergelt, 1969). In the 1980s, Hopfield's energy approach in 1982 (Hopfield, 1982) and the back-propagation learning algorithm for multilayer perceptron's (multilayer feedforward networks) first proposed by Werbos helped renew interest in the field of ANN (Werbos, 1974).

2.2.1 Neural Network Architectures

ANNs can be perceived as weighted directed graphs in which artificial neurons are nodes and directed edges (with weights) are connections between neuron outputs and neuron inputs. They can be grouped into two categories:

- Feed-forward networks, graph has no loops
- Feedback networks, loops occur in graph

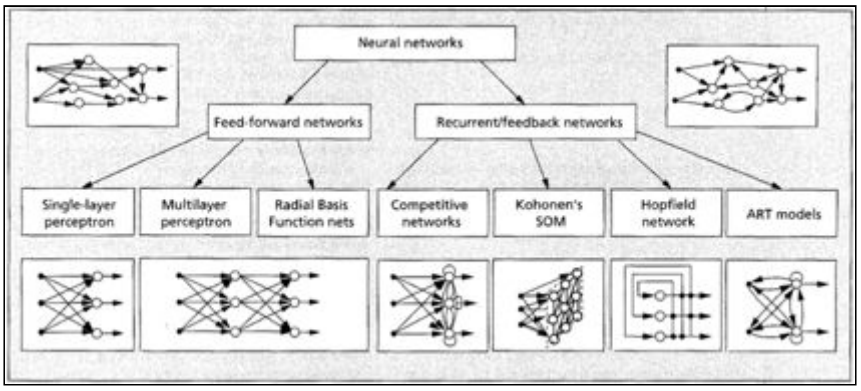


Figure 2.4: An array of feed-forward and feedback network architectures (Jain et al., 1996).

Different architecture yields different network behaviors. In general, feed-forward networks are static, they produce only one set of output values rather than a sequence of values from a given input.

Feed-forward networks require less memory as their response to an input is independent of the previous network state. However, recurrent, or feedback networks are dynamic systems. When a new input pattern is presented, the neuron outputs are computed. Due to the feedback paths, the inputs of each neuron are modified, this leads the network to a new state (Jain et al., 1996).

There are three main learning paradigms: supervised, unsupervised, and hybrid. In supervised learning, the model is provided with a correct answer (output) for every input pattern. Weights are determined to allow the model to produce answers approximate to correct answers.

Reinforcement learning is a variant of supervised learning in which the network is provided with only a critique on the correctness of network outputs, not the correct answers themselves. In contrast, unsupervised learning, does not require a correct answer associated with each input pattern in the training data set. It explores the underlying structure in the data, or relationships between patterns in the data, and organizes patterns into categories from these matchings. Hybrid learning combines supervised and unsupervised learning. Part of the weights are usually determined through supervised learning, while the others are obtained through unsupervised learning.

Most ANNs consists of three layers of processing units: the input layer, hidden layer and outer layer. The units in the input layer are connected to the units in hidden layer, which are then connected to units in the output layer (Chakrabarty, 2010). The figure below showcases the interconnections between the aforementioned layers.

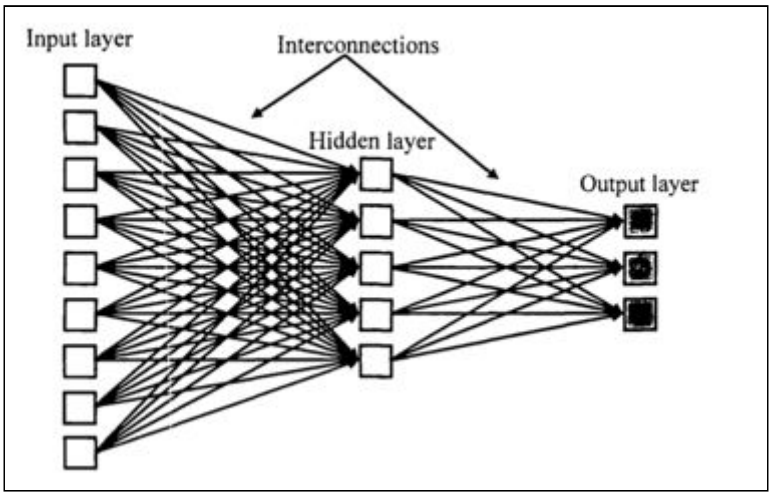


Figure 2.5: An artificial neural network with three layers of processing units (Zakaria et al., 2014).

2.2.2 Training Process

The training process of a neural network consists of defining a cost function and using gradient descent optimization to minimize it.

Mean Squared Error (MSE) is used as the cost function. It can be equated to:

MSE = Sum [(Prediction - Actual)²] * (1 / number of observations)

The MSE works as an estimator (of a procedure for estimating an unobserved quantity) measuring the average of the squares of the errors — that is, the average squared difference between the estimated values and what is estimated. MSE is a risk function, corresponding to the expected value of the squared error loss. MSE results are mostly positive (and not zero) due to randomness or because the estimator does not account for information that could produce a more accurate estimate (Yiu, 2019).

Gradient Descent is the vector of the gradient in a function whose elements are its partial derivatives with respect to each parameter. For example, in attempt to minimize a cost function, C(B0, B1), with just two changeable parameters, B0 and B1, the gradient would be:

Gradient of C(B0, B1) = [[dC/dB0], [dC/dB1]]

With this, each element of the gradient can tell how the cost function would change if a small change to that particular parameter is applied. This makes tweaking the network with approximate values possible.

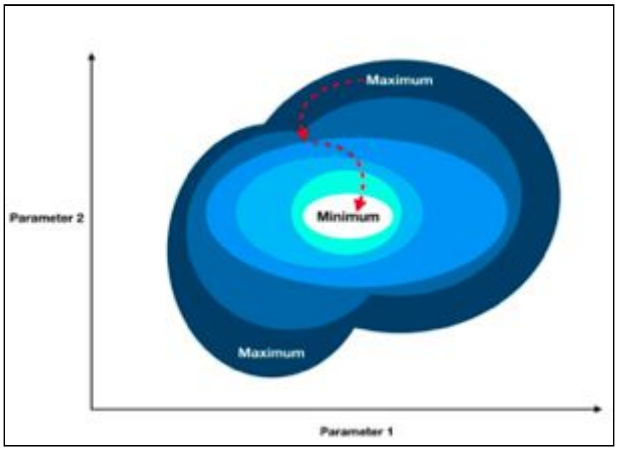


Figure 2.6: Illustration of gradient descent (Yiu, 2019).

To summarize, the small changes can be computed through these steps:

1. Compute the gradient of the "current location" (calculate the gradient using the current parameter values).
2. Modify each parameter by an amount proportional to its gradient element and in the opposite direction of its gradient element. For example, if the partial derivative of the cost function with respect to B0 is positive but small and the partial derivative with respect to B1 is negative and large, then B0 shall be decreased by a tiny amount and increase B1 by a large amount to lower the cost function.
3. Recompute the gradient using the newly tweaked parameter values and repeat the previous steps until the minimum value is reached.

2.3 Convolutional Neural Network (CNN)

In 1989, LeCun invented the convolution of the neural network - LeNet, and its use for digital identification (LeCun et al., 1989). Then in 2012, in response to questions regarding deep learning, Hinton and his students applied deep learning to ImageNet (the largest database in image recognition), and achieved remarkable results (Krizhevsky et al., 2012). CNNs are a variance of neural networks, hierarchical neural network for recognition, particularly images. It works by using significant processes, such as Gradient Descent (Bottou, 2010) and Backpropagation. Furthermore, a typical CNN consists of a sequence of layers, and every layer transform one volume of activations to another through the use of specific functions. These layers consists of the beginning layer, convolution layer, and the output layer. The layers between them are called hidden layers. Then main purpose of convolution layer is to extract image features, then drive them into the hidden layers for computing, and output the results via output layer. Layers among hidden layers usually, such as pooling layers (sub-sampling layers) are partially connected, while the output layers are fully connected (Zhou,

2018). There are several architectures within CNN that are commonly used: AlexNet, LeNet, Inception-v1 (GoogleNet), ResNet, and VGGNet.

2.3.1 Pooling Layers

A pooling layer is added in-between successive Convolutional layers in a CNN architecture. Its function is to gradually reduce the spatial size of the representation to reduce the number of parameters and computation in the network. The Pooling Layer operates independently on every depth slice of the input and resizes it spatially, using the MAX operation (CS231n, n.d.). The most common form is a pooling layer with filters of size 2x2 applied with a stride of 2 down samples every depth slice in the input by 2 along both width and height, discarding 75% (3/4) of the activations. Every MAX operation would in this case be taking a max over 4 numbers (little 2x2 region in some depth slice). However, depth dimension remains unchanged. It works by:

- Accepting a volume of size W1×H1×D1
- Requires two hyperparameters: The spatial extent F, The stride S
- Produces a volume of size W2×H2×D2 where:
    - W2=(W1−F)/S+1
    - H2=(H1−F)/S+1
    - D2=D1

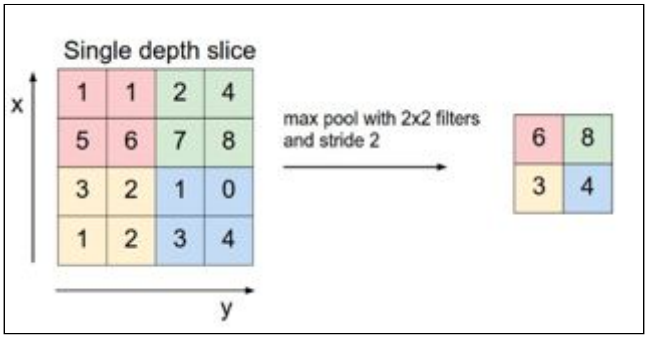


Figure 2.7: Max pooling being performed on a single depth slice; each max is taken over 4 numbers (little 2x2 square) (CS231n, n.d.)

2.3.2 Process

The convolutional layer consists of a set of filters. Each filter is smaller than the input data and the type of multiplication applied between a filter-sized patch of the input and the filter is a dot product resulting in a single value. This method allows the same filter (set of weights) to be multiplied by the input array multiple times at different points on the input.

Specifically, the filter is applied systematically to each overlapping part or filter-sized patch of the input data, left to right, top to bottom. This results in a two-dimensional array of output values that represent a filtering of the input called the "feature map" (Brownlee, 2019).

Successive computational layers alternate between convolutional layers and sampling layers, as spatial resolution decreases in each convolutional layer or sampling layer, the number of feature mappings increases compared to the corresponding previous layer (Zhou, 2018).

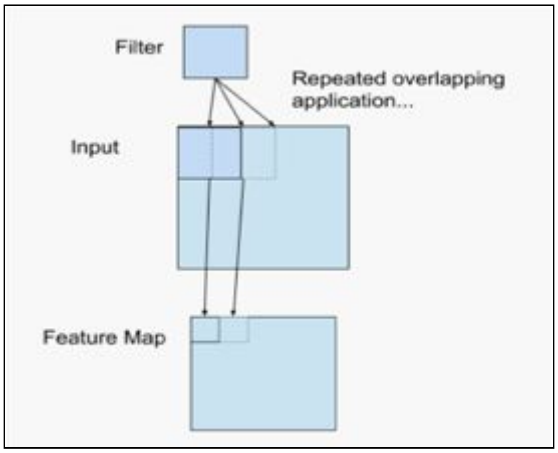


Figure 2.8: Example of a Filter Applied to a Two-Dimensional Input to Create a Feature Map (Brownlee, 2019).

For example, the input layer consists of 32 × 32 sensing nodes, receiving the original image. The calculation then alternates between convolution and sub-sampling, which stated as follows:

The first hidden layer is convoluted, which consists of eight feature maps, each feature map consists of 28 × 28 neurons, each neuron specifies a 5 ×5 acceptance domain; The second hidden layer implements sub-sampling and local averaging. It is also composed of eight feature maps, but each feature map consists of 14 × 14 neurons. Each neuron has a 2 × 2 acceptance field, a training coefficient, a training bias, and a sigmoid activation function. The training factor and the bias control the operating point of the neuron. The third hidden layer is the second convolution, which consists of 20 feature maps, each consisting of 10 × 10 neurons. Each neuron in the hidden layer may have a synaptic connection to several feature maps of the next hidden layer, which operates in a similar manner to the first convolution layer. The fourth hidden layer performs the second sub-sampling and local average calculation. It consists of 20 feature maps, but each feature map consists of 5 x 5 neurons, which operate in a similar manner to the first sample. The fifth hidden layer implements the final stage of convolution, which consists of 120 neurons, each of which specifies a 5 × 5 accepted domains. Finally, a full connection layer, get the output vector.

2.3.3 Backpropagation

Backpropagation is an expression for the partial derivative ∂C/∂w of the cost function C with respect to any weight w (or bias b) in the network (Nielsen, 2019). The expression below is used to determine how quickly the cost changes when the weights and biases are changed.

Definition of the backpropagation error:

$$\frac{\partial E}{\partial b} = \frac{\partial E}{\partial u} \times \frac{\partial u}{\partial b}$$

2.4 CNN Architectures

From 2010 to 2017, The ImageNet Large Scale Visual Recognition Challenge (ILSVRC) evaluates algorithms for object detection and image classification at large scale (Russakovsky et al., 2015). One of their high level of motivation is to allow researchers to compare progress in detection across a wider variety of objects -- taking advantage of the quite expensive labeling effort. Since then. many architectures and methods on image recognition are born, most of them are based on CNN.

2.4.1 AlexNet

In 2012, an architecture used in a research paper titled, “ImageNet Classification with Deep Convolutional Neural Networks” became known as AlexNet due to the first author named Alex Krizhevsky (Krizhevsky et al., 2012). The paper used its proposed framework on CNN to get a Top-5 error rate (rate of not finding the true label of a given image among its top 5 predictions) of 15.3%, which was the lowest rate of that time. AlexNet was the winning entry in ImageNet Large Scale Visual Recognition Challenge (ILSVRC) 2012. It solved the problem of image classification where the input is an image of one of 1000 different classes and the output is a vector of 1000 numbers (Nayak, 2018).

AlexNet utilizes an architecture that is much larger than previous CNNs used in computer vision. It consists of 5 Convolutional Layers and 3 Fully Connected Layers with over 60 million parameters and 650,000 neurons.

The first two Convolutional layers are followed by the Overlapping Max Pooling layers. The third, fourth and fifth convolutional layers are connected directly. The fifth convolutional layer is followed by an Overlapping Max Pooling layer, the output of which goes into a series of two fully connected layers. The second fully connected layer feeds into a SoftMax classifier with 1000 class labels. Lastly, ReLU (Rectified Linear Unit) nonlinearity is applied to the first and second convolution layer followed by a local normalization step before pooling.

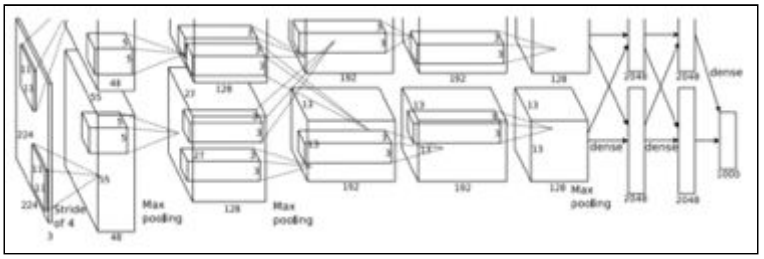


Figure 2.9: AlexNet architecture showing the delegation of responsibilities between the two GPUs. One GPU runs the layer-parts at the top of the figure while the other runs the layer-parts at the bottom (Krizhevsky et al., 2012).

Max Pooling Layers down sample the width and height of the tensors, maintaining the depth. Overlapping Max Pool layers are similar to the Max Pool layers, except the adjacent windows over which the max is computed overlap each other. The authors used pooling windows of size 3×3 with a stride of 2 between the adjacent windows. This overlapping nature of pooling helped reduce the top-1 error rate by 0.4% and top-5 error rate by 0.3% respectively when compared to using non-overlapping pooling windows of size 2×2 with a stride of 2 that would give the same output dimensions.

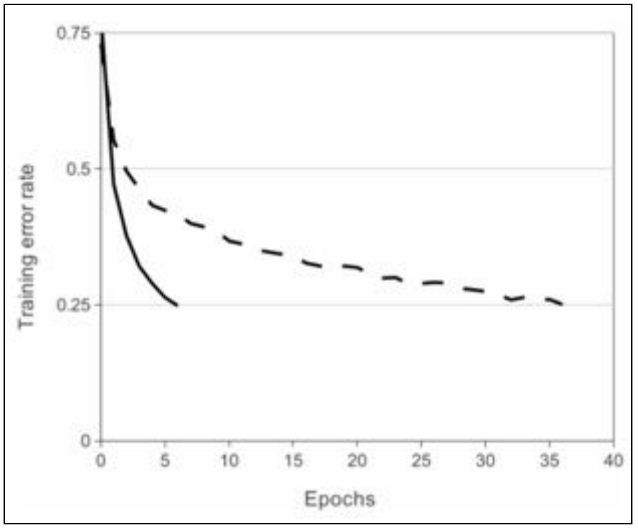


Figure 2.10: A four-layer CNN with ReLUs (solid line) training error rate on CIFAR-10 against tanh neurons (dashed line) (Krizhevsky et al., 2012).

The use of ReLu Nonlinearity allowed deep CNNs to be trained much faster than using the saturating activation functions like tanh or sigmoid. Figure 2.10 shows through the use of ReLUs(solid curve), AlexNet could achieve a 25% training error rate six times faster than an equivalent network using tanh(dotted curve).

With 60 million parameters, the author had unique ways to combat overfitting on image data. One of which is to artificially enlarge the dataset using label-preserving transformations. Two distinct forms of data augmentation were employed, both of which allowed transformed images to be produced from

the original images with minimal computation. The first form of consists of generating image translations and horizontal reflections through extracting g random 224 × 224 patches (and their horizontal reflections) from the 256×256 images and training these images on the network. The second method is to alter the intensities of the RGB channels in training images. The results of the model are summarized in the figure below:

Model	Top-1	Top-5
Sparse coding [2]	*47.1%*	*28.2%*
SIFT + FVs [24]	*45.7%*	*25.7%*
CNN	**37.5%**	**17.0%**

Table 2.1: Comparison of results on ILSVRC2010 test set, CNN using AlexNet framework (Krizhevsky et al., 2012).

2.4.2 VGGNet

VGGNet is invented by VGG (Visual Geometry Group) from the University of Oxford in 2014. It won the 1st runner-up prize of the ILSVRC (ImageNet Large Scale Visual Recognition Competition) 2014 in the classification task and won first place in the localization task category (Simonyan and Zisserman, 2015).

VGGNet utilizes a smaller filter size of 3x3 as opposed to the usual 11x11 in AlexNet. However, by using 3 layers of 3x3 filters, it effectively covers an area of 7x7. This also means that the number of parameters is fewer. Suppose there is only 1 filter per layer and 1 layer at input:

- 1 layer of 11×11 filter, number of parameters = 11×11=121
- 5 layers of 3×3 filters, number of parameters = 3×3×5=45

This results in the total number of parameters to be reduced by 63%. On a larger network, fewer parameters to be learnt produces a faster convergence and reduces the overfitting problem prevalent in older CNN models such as AlexNet.

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

Figure 2.11: VGGNet configurations (In columns). The depth of the configurations increases from the left (A) to the right (E), as more layers are added. The incorporation of 1 × 1 conv. layers (configuration C, Table 1) is a way to increase the nonlinearity of the decision function (Simonyan and Zisserman, 2015).

The architecture utilizes a stack of convolutional layers (which has a different depth in different architectures) is followed by three Fully-Connected (FC) layers: the first two have 4096 channels each, the third performs 1000- way ILSVRC classification and thus contains 1000 channels (one for each class). The final layer is the soft-max layer.

Besides this, all hidden layers are equipped with ReLu non-linearly (Tsang, 2018). Through Figure 2.12, the macro architecture of configuration D of the network (VGG-16) is visualized.

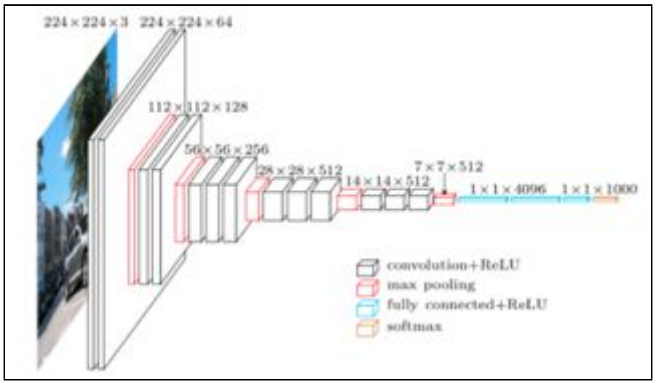


Figure 2.12: Macro Architecture of VGG-16 (Tsang, 2018).

VGG-16 managed to obtained a 9.4% error rate compared the VGG-11 rate of 10.4%, which means the additional three 1×1 conv layer helps improve the classification accuracy. Moreover, it proves that the network improves by adding the number of layers.

One method VGGNet applies to reduce the error rate is to train the network at various scales. For example, in single-scale training an image is scaled with smaller-size equal to 256 or 384, i.e. S=256 or 384, whereas in multi-scale training an image is scaled with smaller-size equal to a range from 256 to 512, i.e. S=[256;512]. Therefore, the range of S is large with multi-scale training. Results shown a reduction in the error rate of VGG-16 from 8.8%/8.7% to 8.1%.

2.4.3 ResNet

ResNet (He et al. 2015), short for Residual Networks is a classic neural network used as a backbone for many computer vision tasks. This model won the ImageNet challenge in 2015. The fundamental breakthrough with allowed training extremely deep neural networks with 150+layers successfully. Prior to ResNet, training very deep neural networks was difficult due to the eventual saturation and the notorious problem of vanishing gradients. The details within the architecture and process behind ResNet will be further explained below.

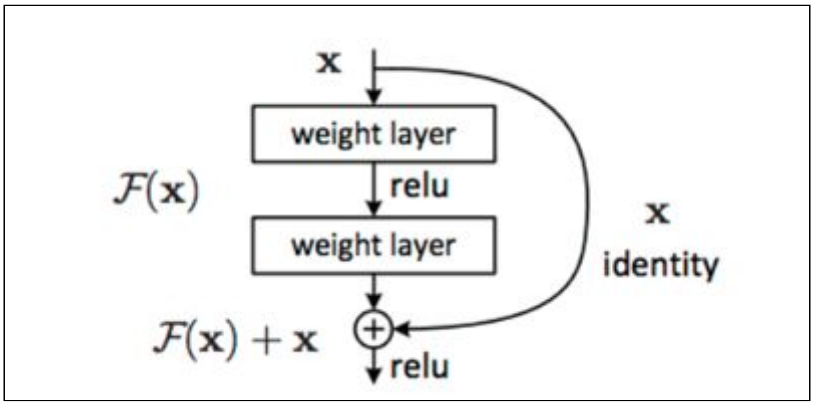


Figure 2.13: Residual learning: a building block (Residual Block) (He et al. 2015).

Figure 2.13 above displays the residual block in action. The main thing to focus here is the identity mapping, X. This identity mapping does not have any parameters and functions to add the output from the previous layer to the layer ahead. However, sometimes x and F(x) will not have the same dimension. Considering that a convolution operation shrinks the spatial resolution of an image, e.g. a 3x3 convolution on a 32 x 32 image results in a 30 x 30 image. The identity mapping is multiplied by a linear projection W (weight) to expand the channels of shortcut to match the residual (Shorten 2019). This allows for the input x and F(x) to be combined as input to the next layer.

Equation used when F(x) and x have a different dimensionality:

$$y = F(x, \{W_i\}) + W_s x$$

ResNet consists of one convolution and pooling step followed by 4 layers of the same pattern. They perform 3x3 convolution with a fixed feature map dimension [64, 128, 256, 512], bypassing the input

every 2 convolutions using identity mapping. Moreover, the width (W) and height (H) dimensions remain constant during the entire layer.

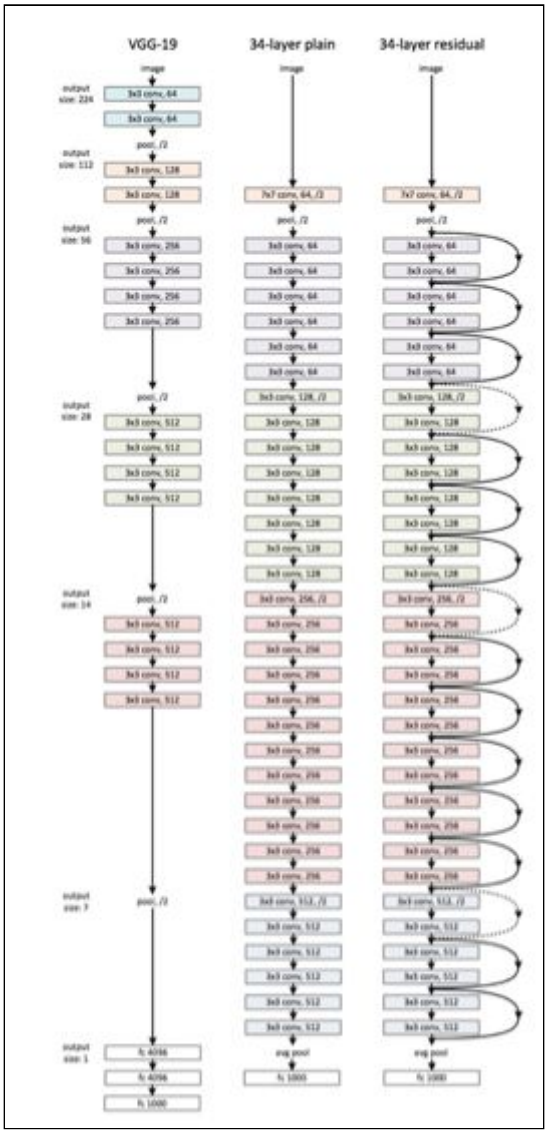


Figure 2.14: Example network architectures for ImageNet. **Left**: the VGG-19 model as a reference. **Middle**: a plain network with 34 parameter layers. **Right**: a residual network with 34 parameter layers. The dotted shortcuts increase dimensions (He et al. 2015).

According to Figure 2.14 above, the dotted line represents a change in the dimension of the input volume (reduction). This reduction between layers is achieved by an increase on the stride, from 1 to 2, at the first convolution of each layer; instead of by a pooling operation. The image below is a summary of output size at individual layers and the dimensions of the convolutional filters at every point.

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
conv2_x	56×56	3×3 max pool, stride 2 $\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix} \times 2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

Figure 2.15: ResNet Architectures in different layers. Building blocks are shown in brackets with the numbers of blocks stacked. Downsampling is performed by conv3_1, conv4_1, and conv5_1 with a stride of 2 (He et al. 2015).

2.5 Object Detection CNN Architectures

Computer vision is an interdisciplinary field that has been gaining huge amounts of traction in recent years(since CNN) and self-driving cars have taken centre stage. Another integral part of computer vision is object detection. Object detection aids in pose estimation, vehicle detection, surveillance etc. The difference between object detection algorithms and classification algorithms is that in detection algorithms, where a bounding box are drawn around the object of interest to locate it within the image. Also, it is not necessary just one bounding box is drawn in an object detection case, there could be many bounding boxes representing different objects of interest within the image.

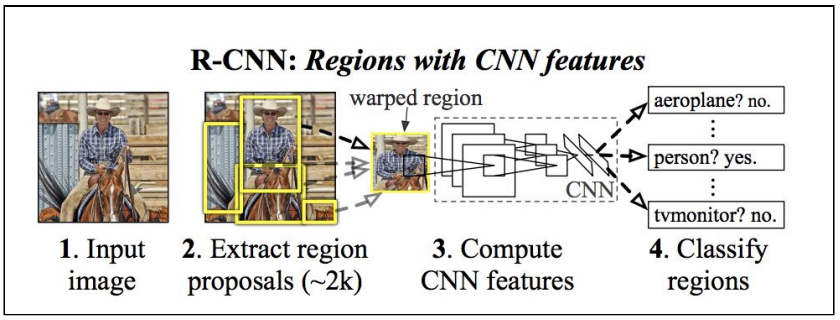


Figure 2.16: R-CNN architecture (Girshick et al., 2014).

The major reason a standard CNN cannot proceed with this problem is that, the length of the output layer is variable — not constant, this is because the number of occurrences of the objects of interest is not fixed. A naive approach to solve this problem would be to take different regions of interest from the image, and use a CNN to classify the presence of the object within that region. The problem with this approach is that the objects of interest might have different spatial locations within the image and different aspect ratios. Hence, a huge number of regions is needed to be selected and this could computationally blow up. Therefore, algorithms like Region CNN (R-CNN) (Girshick et al., 2014), YOLO (Redmon et al., 2016) etc have been developed to find these occurrences and find them fast.

2.5.1 R-CNN

To bypass the problem of selecting a huge number of regions, Girshick et al. (2014) proposed a method where selective search is used to extract just 2000 regions from the image and was called region proposals. Therefore, instead of trying to classify a huge number of regions, only 2000 regions are being worked on. These 2000 region proposals are generated using this selective search algorithm:

1. Generate initial sub-segmentation by generating many candidate regions.
2. Use greedy algorithm to recursively combine similar regions into larger ones.
3. Use the generated regions to produce the final candidate region proposals.

These 2000 candidate region proposals are warped into a square and fed into a CNN that produces a 4096-dimensional feature vector as output. The CNN acts as a feature extractor and the output dense layer consists of the features extracted from the image and the extracted features are fed into a Support Vector Machine (SVM) to classify the presence of the object within that candidate region proposal. In addition to predicting the presence of an object within the region proposals, the algorithm also predicts four values which are offset values to increase the precision of the bounding box. For example, given a region proposal, the algorithm would have predicted the presence of a person but the face of that person within that region proposal could've been cut in half. Therefore, the offset values help in adjusting the bounding box of the region proposal.

Problems with R-CNN:

- It still takes a huge amount of time to train the network as the neural network would have to classify 2000 region proposals per image.
- It cannot be implemented in real time as it takes around 47 seconds for each test image.
- The selective search algorithm is a fixed algorithm. Therefore, no learning is happening at that stage. This could lead to the generation of bad candidate region proposals.

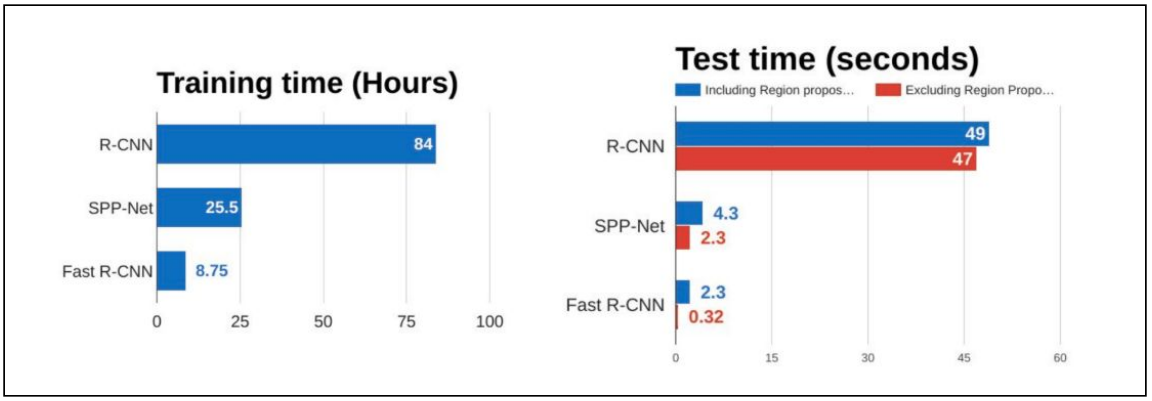


Figure 2.17: Fast R-CNN is significantly faster in training and testing sessions over R-CNN. The performance of Fast R-CNN during testing time, including region proposals slows down the algorithm significantly when compared to not using region proposals. Therefore, region proposals become bottlenecks in Fast R-CNN algorithm affecting its performance (John et al., 2019).

2.5.2 Fast R-CNN

The same team (Girshick et al., 2015) solved some of the drawbacks of R-CNN to build a faster object detection algorithm and it was called Fast R-CNN. The approach is similar to the R-CNN algorithm. But, instead of feeding the region proposals to the CNN, input image is fed into the CNN to generate a convolutional feature map. From the convolutional feature map, the region of proposals is identified and wrapped into squares, and by using a RoI pooling layer, the region of proposals are reshaped into a fixed size so that it can be fed into a fully connected layer. From the RoI feature vector, a softmax layer is used to predict the class of the proposed region and also the offset values for the bounding box.

The reason "Fast R-CNN" is faster than R-CNN is because the 2000 region proposals are not needed to be fed into the CNN every time. Instead, the convolution operation is done only once per image and a feature map is generated from it.

2.5.3 YOLO - You Only Look Once

All of the previous object detection algorithms use regions to localize the object within the image. The network does not look at the complete image. Instead, parts of the image which have high probabilities of containing the object. YOLO or You Only Look Once (Redmon et al., 2016) is an object detection algorithm much different from the region based algorithms seen above. In YOLO a single CNN predicts the bounding boxes and the class probabilities for these boxes.

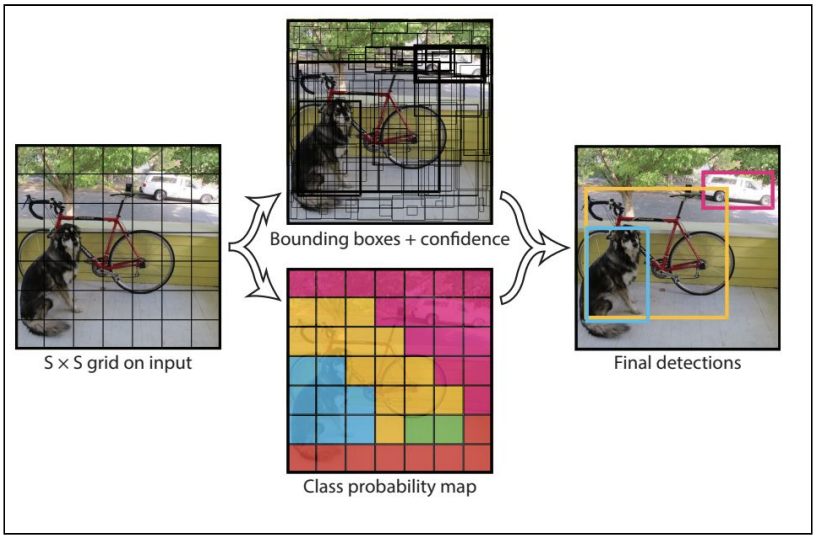


Figure 2.18: YOLO models detection as a regression problem. It divides the image into an S × S grid and for each grid cell predicts B bounding boxes, confidence for those boxes, and C class probabilities. These predictions are encoded as an S × S × (B ∗ 5 + C) tensor (Redmon et al., 2016).

YOLO works by taking an image and splitting it into an SxS grid, within each of the grid it takes m bounding boxes. For each of the bounding box, the network outputs a class probability and offset values for the bounding box. The bounding boxes having the class probability above a threshold value is selected and used to locate the object within the image.

YOLO is orders of magnitude faster (45 frames per second) than other object detection algorithms. The limitation of YOLO algorithm is that it struggles with small objects within the image, for example it might have difficulties in detecting a flock of birds. This is due to the spatial constraints of the algorithm.

2.6 Conclusion

From this chapter, it is found that patterns can be a set of measurements or observations, usually represented in vector notation, whereas features are the measurements extracted from the patterns. And for pattern recognition, there are statistical and structural approaches. On the other hand, neural networks are a set of algorithms, modeled loosely after the human brain, that are designed to recognize patterns and CNNs are a variance of neural networks, hierarchical neural network for recognition, particularly images. Besides, being motivated by the ImageNet Large Scale Visual Recognition Challenge, researchers have created many CNN architectures to compete against other architectures and make breakthroughs and improvements. Finally, based on the CNN architectures, R-CNNs are created and designed for object detection.

Chapter 3

Research Methodology

3 Research Methodology

In this chapter, the methodology of this research shall be revealed. Topics such as the proposed framework, dataset and evaluation method shall be discussed. These subsets of the methodology lay in the scope of practicality and real world implementations that is also backed by their theories.

3.1 Proposed Framework

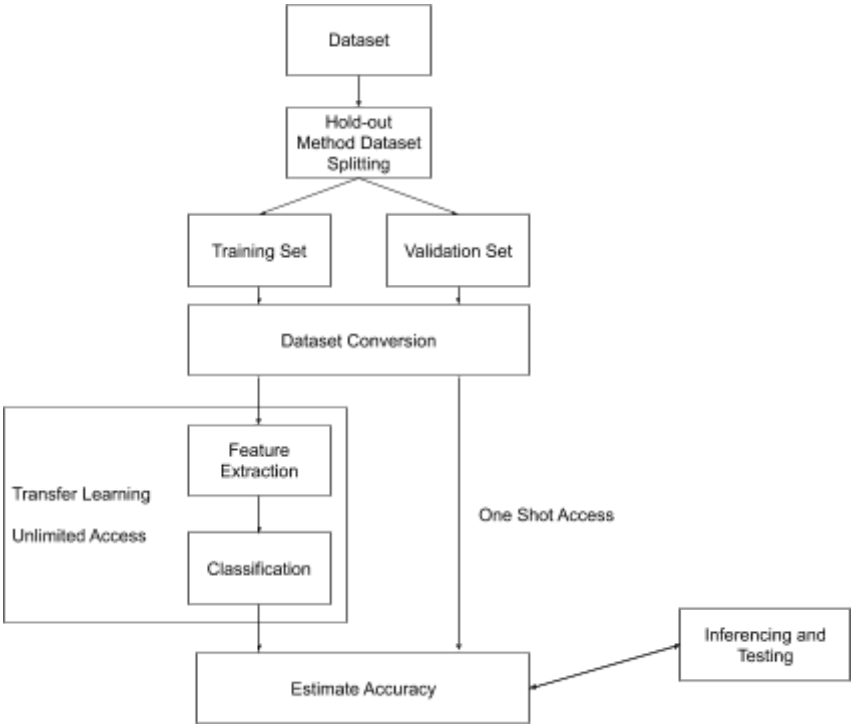


Figure 3.1: Flow of proposed framework.

Figure 3.1 illustrates a straightforward flow diagram of the proposed framework which can be summarized as follows:

1. Dataset:

   DeepFashion (Liu et al., 2016) is chosen as the dataset. It is acquired from the Multimedia Laboratory of Chinese University of Hong Kong. It consists of a total number of 800,000 annotated images, but for clothing image retrieval, there are only 300,000 pictures available. This dataset can only be used for non-commercial research purposes.

2. Dataset Splitting:

   The holdout method during the evaluation phase requires the dataset to be split into a training set, and a validation set. The DeepFashion team has prepared these two sets, which the format consists of 250,000 pictures, and the latter has 50,000 pictures.

3. Dataset Conversion:

   FashionNet (Liu et al., 2016) - a VGG-16 based architecture used for the original DeepFashion paper uses a non-standard format of annotation and labels for all images, which are also stored in multiple text files. By converting this dataset to the TFRecord format that is supported by TensorFlow (Abadi et al., 2016), a greater degree of flexibility is obtained as most functions offered by TensorFlow is now available. TFRecord also improves learning performance, as well as a slight gain in accuracy of the final model.

4. Transfer Learning and Evaluation:

   A FasterRCNN-Inception-v2 model pre-trained on the COCO (Lin et al., 2014) dataset is used for transfer learning. This phase is split into two parts - feature extraction and classification. Faster R-CNN (Szegedy et al., 2015) hosts extra layers on the Inception-v2 (Szegedy et al., 2016) base model, which hugely improves object detection. The training set has unlimited access to the neural network (feature layers and classifiers). Whereas the validation set only has one shot access to estimate the accuracy of the network, which is part of the evaluation phase. TensorFlow Object Detection API is used as the base framework for all the tasks mentioned here, whereas TensorBoard is used to visualize real-time results.

5. Inferencing and Testing:

   Inferencing and testing on both still image and motion image files are provided built-in by the TensorFlow Object Detection API. If a live video feed is required, e.g. external camera or webcam, OpenCV (Bradski, 2000) can be used.

A large magnitude of this proposed framework is common among many image classification tasks, especially the part about transfer learning. The hold-out method is also commonly used to avoid overfitting. However, dataset conversion is not typically performed; yet the increased performance that it potentially provide is found to be beneficial, more of this is explained in Section 3.1.1.

The flexibility and performance TensorFlow provides is optimal for the needs of this research when choosing between many others such as pytorch (Paszke et al., 2017), and Keras (Chollet, 2015). TensorFlow 2.0 is still in beta stage and is lacking for several sub-frameworks such as Faster R-CNN (Ren et al., 2017) (TensorFlow, 2019), as such, TensorFlow 2.0 is not considered. Details about TensorFlow and Faster R-CNN is revealed in Section 3.1.2.

3.1.1 Dataset Conversion

DeepFashion (Liu et al., 2016) is a rich dataset which can fulfill all the needs for image classification tasks. However, it is also made and catered for image classification framework of that time, namely FashionNet (Liu et al., 2016) which is based on VGG-16 pretrained model.

The annotation data in DeepFashion is rather unusually arranged and can be rather inefficient in terms of preprocessing work needed before used for training. For instance, all annotation data in DeepFashion, as well as labels of training, validation, and testing image, are all written in multiple text files. This in turn requires extensive time to implement other frameworks not designed for the dataset.

As such, converting the dataset to TFRecord format can standardize this dataset for multiple use cases based on the TensorFlow framework. The TFRecord format can also help improve performance as it can store thousands or millions of files into one single file or set of files in serialized form. The serialization is especially important when the dataset is streamed over a network for cross-domain training and processing. There are guidelines on the TensorFlow tutorials on how to convert datasets to TFRecord format.

For this research, the dataset has been converted from hundreds of thousands of image files to three TFRecord formatted files, namely for training, validation, and testing, respectively.

3.1.2 Faster R-CNN on TensorFlow

Faster R-CNN is a model built for efficient object detection, which includes multiple classification and localization outputs. It adds extra layers on a base model such as Inception-v2 to improve performance for object detection. It was originally built for Caffe (Jia et al., 2014) but alternate implementations exist. One of the many alternate implementations of Faster R-CNN is the Object Detection API of TensorFlow framework, though many other choices of object instance segmentation frameworks are also available. Historically, Faster R-CNN is built to provide better performance compared to Fast R-CNN (Girshick, 2015).

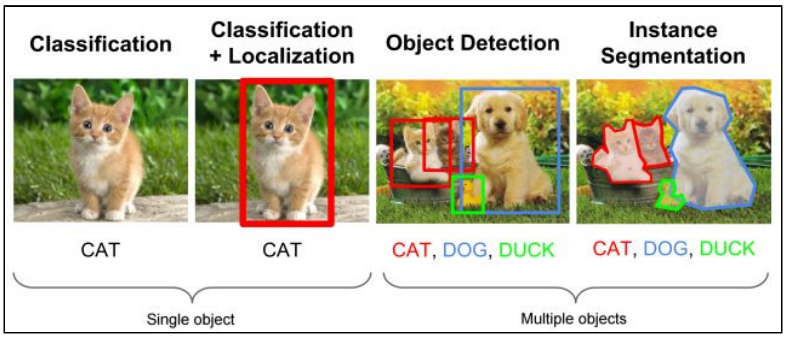


Figure 3.5: Differences between different image classification output method (Quaknine, 2018).

3.1.3 Considerations Against Mask R-CNN

Mask R-CNN (He et al., 2017) is a state of the art model built for object instance segmentation tasks and a new masking capability in replacement to bounding boxes. Mask annotation is a prerequisite in using this framework, which DeepFashion lacks. It is therefore not considered for this study.

3.1.4 Transfer Learning on FasterRCNN-Inception-v2 using TensorFlow

TensorFlow is a robust solution for many machine learning tasks, including transfer learning. For this research. FasterRCNN-Inception-v2 pre-trained model is chosen as a base model for its efficiency and excellent accuracy. This model is pre-trained on the COCO (Lin et al., 2014) dataset which is a large dataset consists of 328,000 images. Configurations and hyper-parameters catered for this model is provided by the object detection API of TensorFlow, TensorFlow allows instances of training and evaluation steps to be performed automatically.

On fine-tuning for transfer learning, since DeepFashion is a large dataset but its features to extract are different from the original COCO dataset, only the architecture and the weights of the FasterRCNN-Inception-v2 model is used for transfer learning, hence, its previous knowledge fron the COCO dataset is discarded (Marcelino, 2018).

TensorBoard is used to illustrate the learning process in real time. The learning instances is set to run indefinitely until manually terminated. Criteria of termination is based on the scalar accuracy figures provided by TensorBoard. Those figures are observed to determine the state of the learning process; whether its accuracy/error has reached a global maxima/minima, whether it has converged, or even when overfitting starts to occur.

3.1.5 Inferencing and Testing

Object detection API from TensorFlow is catered for motion image inferencing. FasterRCNN-Inception-V2 is measured to be able to run each frame at about 58ms, which is about 17 frames per second on the COCO dataset. Live feed can also be implemented through OpenCV, with simultaneous processing over the live feed data and real time output with object detection boxes and labels. However, the API can also be used for simple still image inferencing.

Inference graphs are also generated periodically during the learning phase, thus inferencing is also performed simultaneously with learning for data generation on TensorBoard. With the right configuration, image or video inferencing can also be performed periodically during the learning phase.

Model name	Speed (ms)	COCO mAP[^1]	Outputs
ssd_mobilenet_v1_coco	30	21	Boxes
ssd_mobilenet_v1_0.75_depth_coco ☆	26	18	Boxes
ssd_mobilenet_v1_quantized_coco ☆	29	18	Boxes
ssd_mobilenet_v1_0.75_depth_quantized_coco ☆	29	16	Boxes
ssd_mobilenet_v1_ppn_coco ☆	26	20	Boxes
ssd_mobilenet_v1_fpn_coco ☆	56	32	Boxes
ssd_resnet_50_fpn_coco ☆	76	35	Boxes
ssd_mobilenet_v2_coco	31	22	Boxes
ssd_mobilenet_v2_quantized_coco	29	22	Boxes
ssdlite_mobilenet_v2_coco	27	22	Boxes
ssd_inception_v2_coco	42	24	Boxes
faster_rcnn_inception_v2_coco	58	28	Boxes
faster_rcnn_resnet50_coco	89	30	Boxes
faster_rcnn_resnet50_lowproposals_coco	64		Boxes
rfcn_resnet101_coco	92	30	Boxes
faster_rcnn_resnet101_coco	106	32	Boxes
faster_rcnn_resnet101_lowproposals_coco	82		Boxes
faster_rcnn_inception_resnet_v2_atrous_coco	620	37	Boxes
faster_rcnn_inception_resnet_v2_atrous_lowproposals_coco	241		Boxes
faster_rcnn_nas	1833	43	Boxes

Figure 3.6: Differences in speed of different pre-trained models used on different object detection frameworks (TensorFlow, 2019).

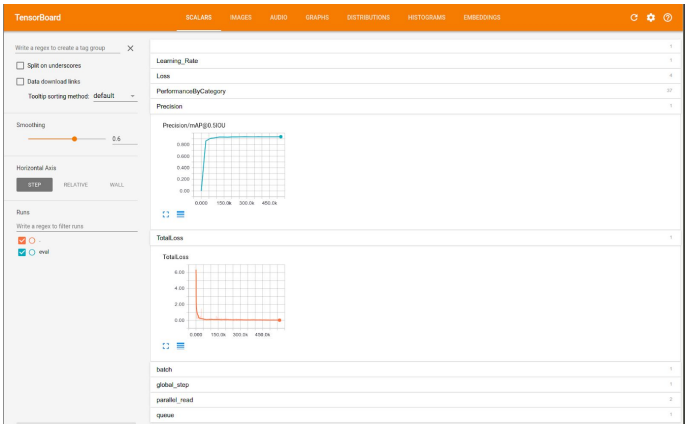


Figure 3.7: Real-time graph generation on TensorBoard (TensorFlow, 2019).

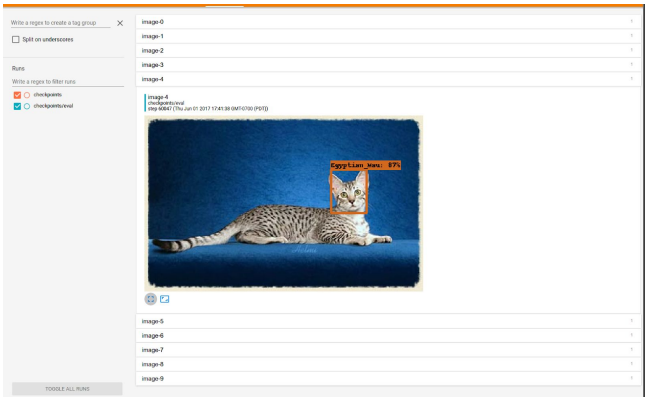


Figure 3.8: Real-time image inferencing on TensorBoard (TensorFlow, 2019).

3.2 Dataset

The adoption of clothes recognition algorithms in real world applications often faces three fundamental challenges (Kiapour et al., 2015). First, the large variations in style, texture, and cutting of clothes tend to confuse many systems. Second, deformation and occlusion are common among clothing items. Third, serious variations are often exhibited in clothing images when they are taken under different scenarios, e.g. a selfie photo versus an online shopping photo of the same product.

Attempts by many studies to handle the above challenges focus on the annotation of the clothes datasets either with semantic attributes (Chen et al., 2012), clothing locations (Luo et al., 2013), or cross-domain image correspondences (similarities between shop images and street images) (Huang et al., 2015). However, none of the aforementioned datasets contain multiple types of annotation.

3.2.1 DeepFashion

DeepFashion (Liu et al., 2016) contains these multiple types of annotation and the research team has claimed that clothes recognition can be benefited from learning these annotations jointly. The ultimate description of DeepFashion as given by the researchers was "a comprehensively annotated clothes dataset that contains massive attributes, clothing landmarks, as well as cross-pose/cross-domain correspondences of clothing pairs".

Figure 3.9: DeepFashion features 50 categories and 1,000 attributes for attonations of clothing images; the attributes consists of five groups: texture, fabric, shape, part, and style (Liu et al., 2016).

DeepFashion has several advantages over most other datasets:

1. Comprehensiveness - each image in DeepFashion is annotated with category, attributes, landmarks, and its cross-pose/cross-domain pair. It has 50 categories and 1,000 attributes. The attributes consists of five groups: texture, fabric, shape, part, and style. The landmarks are also grouped to upper-body, lower-body and full body items.
2. Scale - DeepFashion contains over 800,000 annotated clothing images, far richer than many other datasets. This scale tends to reduce risk of overfitting.

3. Availability - DeepFashion is made public by the research team for the benefit of all research regarding fashion recognition and retrieval.

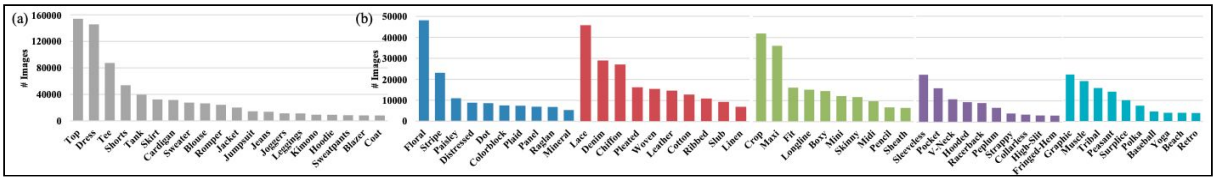


Figure 3.10: Distribution of images of the top-20 categories and top-10 attributes in each attribute group (Liu et al., 2016).

Figure 3.11: Landmarks and pair annotation in DeepFashion (Liu et al., 2016).

3.2.2 Considerations Against DeepFashion2

Figure 3.12: From (a), DeepFashion only has one item per image, annotated with 4-8 sparse landmarks. The bounding boxes are estimated from the landmarks, which cause them to be noisy. In (b), each image can have one to seven items. Each item is manually labeled with bounding box, mask, and dense landmarks (20 average per item) (Ge et al., 2019).

DeepFashion2 (Ge et al., 2019) is published recently to resolve several weaknesses of the original DeepFashion (Liu et al., 2016), among them are:

1. Single item per image
2. Sparse landmarks (4-8 only)
3. No per-pixel masks

Though DeepFashion2 is even more comprehensive than DeepFashion in terms of its annotation data, it is ultimately not chosen in this research due to its limited number of clothing categories compared to its predecessor (13 versus 50). The number of category labels is important for the evaluation work of formality of clothing, as more category labels would be able to provide a more robust formality rating.

3.3 Evaluation Method

Two prominent evaluation methods are holdout method and cross-validation (Kohavi, 2001). The holdout method can be used as a large dataset is chosen for this research. More about this method is discussed in Section 3.3.1.

In Section 3.3.2, method to evaluate classificated images is revealed.

3.3.1 Holdout

The holdout method, sometimes called test sample estimation, partitions the data into two mutually exclusive subsets called a training set and a test set, or holdout set (Kohavi, 2001). It is common to designate 2/3 of the data as the training set and the remaining 1/3 as the validation set. The training set is given to the inducer, an induction algorithm, and the induced classifier is tested on the validation set. Formally, let D_h, the holdout set, be a subset of D of size h, and let D_t be D \ D_h.

The holdout estimated accuracy is defined in this formula:

$$acc_h = \frac{1}{h} \sum_{(v_i, Y_i) \in D_h} \delta(L(D_t, v_i), Y_i), where\ \delta(i;j) = 1\ if\ i = j\ and\ 0\ otherwise$$

Assuming that the inducer's accuracy increases as more instances are seen, the holdout method is a pessimistic estimator because only a portion of the data is given to the inducer for training. The more instances is left for the validation set, the higher the bias of estimate; however, fewer validation set instances means that the confidence interval for the accuracy will be wider. The training set is also advised to be shuffled to avoid overfitting.

To translate these theories in practical usage, the dataset is split into training set and validation set, in case for DeepFashion 1/6 of the dataset is already pre-partitioned as validation set. The shuffling of

the dataset is performed when the dataset is being converted into the TFRecord format, thus able to reduce overfitting.

3.3.2 Evaluation of Classification

The main purpose of performing evaluation is to compare the accuracy of the training set against data that is not present in that set (Skalski, 2018). The mere output of the training set is mostly biased and overfitted. As discussed in Section 3.3.1, the dataset is split into two parts. The validation set is an excellent set of data to reduce the biases of the classification.

To obtain more data is the most preferred way to prevent overfitting. Fortunately, DeepFashion is rich enough to reduce this risk. However, there are still risks that a dataset might caused the neural network to have high variance. Ergo, the tensor is configured to apply regularization. Regularization involves adding an extra element to the loss function, which punishes the model for being too complex or, in simple words, for using too high values in the weight matrix. This way, its flexibility is limited, but is also encouraged to build solutions based on multiple features. Two popular versions of this method are L1 - Least Absolute Deviations (LAD) and L2 - Least Square Errors (LS).

Equations that describe L1 and L2 regularizers:

$$J_{L1}(W,b) = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}^{i}, y^{i}) + \lambda \left\| W \right\|_1, \qquad \left\| W \right\|_1 = \sum_{j=1}^{n_x} |W_j|$$

$$J_{L2}(W,b) = \frac{1}{m}\sum_{i=1}^{m} L(\hat{y}^{i}, y^{i}) + \lambda \left\| W \right\|_2, \qquad \left\| W \right\|_2 = \sum_{j=1}^{n_x} W_j{}^2$$

In practical, during the evaluation phase, the L1 regularizer is used because it reduces the weight values of less important features to zero, very often eliminating them completely from the calculations. In a way, it is a built-in mechanism for automatic feature selection. Moreover, L2 does not perform very well on datasets with a large number of outliers and large number of features. The DeepFashion dataset contains many annotations and street domain images, thus, the L2 regularizer is likely to perform poorly.

3.4 Summary

There exists many methods in various levels of the process of image classification and pattern recognition that aims to achieve optimal results within known use cases. The generic levels of the process would be: First, the methods in obtaining the input dataset for the image classification tasks - either still images or motion images. Second, the methods used for image classification such as the framework used - TensorFlow (Abadi et al., 2016), pytorch (Paszke et al., 2017), Keras (Chollet, 2015), etc. Third, whether a pre-trained model is used, if so, which pre-trained model - Resnet-50 (He et al., 2015), VGG-16 (Simonyan and Zisserman, 2015), Inception-v2 (Szegedy et al., 2016), etc. Forth, the framework or tools used to output the results - bounding box, masking, labelling, or plain text output. And finally, the methods and approaches in evaluating the results - whether the holdout method, or the cross-validation method, and to choose between the L1 and L2 regularizers.

All of these choices of methods, combined with careful configuration, makes up a framework on neural network that is able to serve required needs.

Chapter 4

Theory Background

4 Theory Background

Theoretical research upon Inception-v1 and Inception-v2 - the chosen model, and transfer learning is presented in this chapter. For the Inception model architectures, 1x1 convolutions, global average pooling, and the Inception module is explained. Differences between Inception-v1 and Inception-v2 are also detailed in this chapter.

4.1 Inception-v1

Inception-v1 a.k.a. GoogLeNet (Szegedy et al. 2014) was the winner of the ILSVRC 2014 competition, achieving a top-5 error rate of 6.67%. The architecture was built upon an existing network, LeNet-5 (Lecun et al. 1998), thus "LeNet" is contained within the name. It consisted of a 22-layer deep CNN but reduced the number of parameters from 60 million of AlexNet (Krizhevsky et al. 2012) to 4 million.

Some issues the Faster R-CNN is aimed to address:

- Important parts in the image can have extremely large variation in size. For instance, a set of images with a dog, the area occupied by the dog is different in each image.
- Because of this huge variation in the location of the information, choosing the right kernel size for the convolution operation becomes tough. A larger kernel is preferred for information that is distributed more globally, and a smaller kernel is preferred for information that is distributed more locally.
- Very deep networks like VGGNet (Simonyan and Zisserman 2015) are prone to overfitting. It also hard to pass gradient updates through the entire network.
- Naively stacking large convolution operations is computationally expensive.

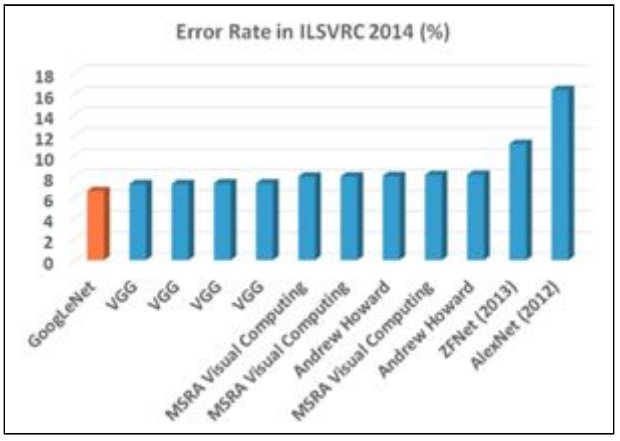


Figure 4.1: ILSVRC 2014 Error Rate (%). Inception-v1 (GoogLeNet) having the lowest rate at 6.67%, followed by VGG at 7.32% (Tsang, 2018).

Inception-v1 architecture contains various new techniques such as 1 x 1 Convolutions, global average pooling and the implementation of a new module called Inception module. This new module works by stacking multiple outputs of the same input through different sizes/types of convolutions (Tsang 2018).

1) The 1x1 Convolutions
    - Originally introduced by Network in Network (Lin et al. 2013).
    - Used as a dimension reduction module to reduce the computation. By reducing the computation bottleneck, depth and width can be increased.

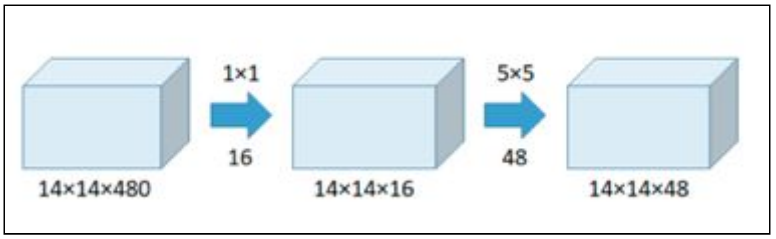


Figure 4.2: Example of 1x1 convolutions followed by a 5x5 convolution using a 14x14x480 input (Tsang 2018).

Through the use of Figure 4.2 above, the total number of operations needed is computed by:

Number of operations for 1×1 = (14×14×16) × (1×1×480) = 1.5M

Number of operations for 5×5 = (14×14×48) × (5×5×16) = 3.8M

Total number of operations = 1.5M + 3.8M = 5.3M

As opposed to performing 5x5 convolutions without the use of 1x1 convolution:

Number of operations = (14×14×48) × (5×5×480) = 112.9M (Compared to 5.3M from added a 1x1 convolution)

In conclusion, using a 1x1 convolutions allows the inception module to be built without increasing the number of operations, which in turn reduces model size.

2) Global average pooling
    - In Inception-v1 (GoogLeNet), global average pooling is used almost at the end of network by averaging each feature map from 7×7 to 1×1
    - Previously, fully connected (FC) layers are used at the end of network, such as in AlexNet. All inputs are connected to each output.
    - The authors found that a move from fully connected layers to average pooling improved the top-1 accuracy by about 0.6%.

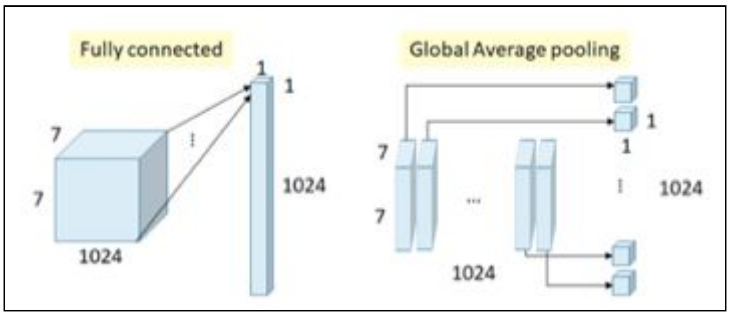


Figure 4.3: Fully Connected Layer VS Global Average Pooling (Tsang 2018).

3) Inception Module

- The Inception module works through the use of parallel paths with different receptive field sizes and operations are meant to capture sparse patterns of correlations in the stack of feature maps.
- Uses 1x1 convolutions for dimensionality reduction before expensive convolutions.

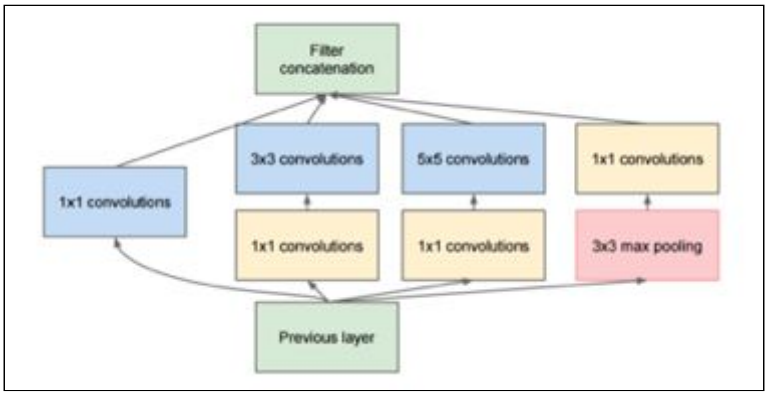


Figure 4.4: : Inception module with dimensionality reduction (1x1 convolution added) (Tsang 2018).

The network was designed with computational efficiency and practicality in mind, so that inference can be run on individual devices including even those with limited computational resources, especially with low-memory footprint. The network is 22 layers deep when counting only layers with parameters (or 27 layers including pooling). The overall number of layers (independent building blocks) used for the construction of the network is about 100 (Szegedy et al. 2014).

type	patch size/ stride	output size	depth	#1×1	#3×3 reduce	#3×3	#5×5 reduce	#5×5	pool proj	params	ops
convolution	7×7/2	112×112×64	1							2.7K	34M
max pool	3×3/2	56×56×64	0								
convolution	3×3/1	56×56×192	2		64	192				112K	360M
max pool	3×3/2	28×28×192	0								
inception (3a)		28×28×256	2	64	96	128	16	32	32	159K	128M
inception (3b)		28×28×480	2	128	128	192	32	96	64	380K	304M
max pool	3×3/2	14×14×480	0								
inception (4a)		14×14×512	2	192	96	208	16	48	64	364K	73M
inception (4b)		14×14×512	2	160	112	224	24	64	64	437K	88M
inception (4c)		14×14×512	2	128	128	256	24	64	64	463K	100M
inception (4d)		14×14×528	2	112	144	288	32	64	64	580K	119M
inception (4e)		14×14×832	2	256	160	320	32	128	128	840K	170M
max pool	3×3/2	7×7×832	0								
inception (5a)		7×7×832	2	256	160	320	32	128	128	1072K	54M
inception (5b)		7×7×1024	2	384	192	384	48	128	128	1388K	71M
avg pool	7×7/1	1×1×1024	0								
dropout (40%)		1×1×1024	0								
linear		1×1×1000	1							1000K	1M
softmax		1×1×1000	0								

Figure 4.5: Inception-v1 (GoogLeNet) incarnation of the Faster R-CNN (Szegedy et al. 2014).

4.2 Inception-v2

Inception-v2 (Szegedy et al. 2016), the chosen model for this research, is proposed by the same team from Inception-v1 the following year with a number of upgrades which increased the accuracy and reduced the computational complexity, issues that were addressed to solve were:

- Reduce representational bottleneck. The intuition was that, neural networks perform better when convolutions didn't alter the dimensions of the input drastically. Reducing the dimensions too much may cause loss of information, known as a "representational bottleneck".
- Using smart factorization methods, convolutions can be made more efficient in terms of computational complexity.

A few solutions were proposed, they are:

- Factorize 5x5 convolution to two 3x3 convolution operations to improve computational speed. Although this may seem counterintuitive, a 5x5 convolution is 2.78 times more expensive than a 3x3 convolution. So stacking two 3x3 convolutions in fact leads to a boost in performance. This is illustrated in the below image.

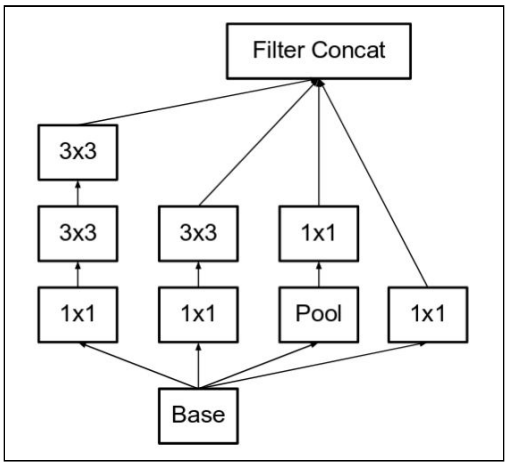


Figure 4.6: The left-most 5x5 convolution of Inception-v1 (refer Figure 4.4) is now represented as two 3x3 convolutions (Szegedy et al. 2016).

- Convolutions of filter size nxn are factorized to a combination of 1xn and nx1 convolutions. For example, a 3x3 convolution is equivalent to first performing a 1x3 convolution, and then performing a 3x1 convolution on its output. They found this method to be 33% cheaper than the single 3x3 convolution. This is illustrated in Figure 4.7:

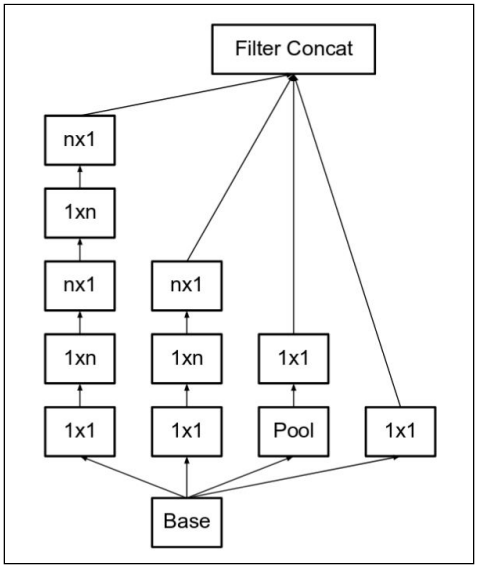


Figure 4.7: Here, put n=3 to obtain the equivalent of the Figure 4.6. The left-most 5x5 convolution can be represented as two 3x3 convolutions, which inturn are represented as 1x3 and 3x1 in series (Szegedy et al. 2016).

- The filter banks in the module were expanded (made wider instead of deeper) to remove the representational bottleneck. If the module was made deeper instead, there would be excessive reduction in dimensions, and hence loss of information. This is illustrated in Figure 4.8:

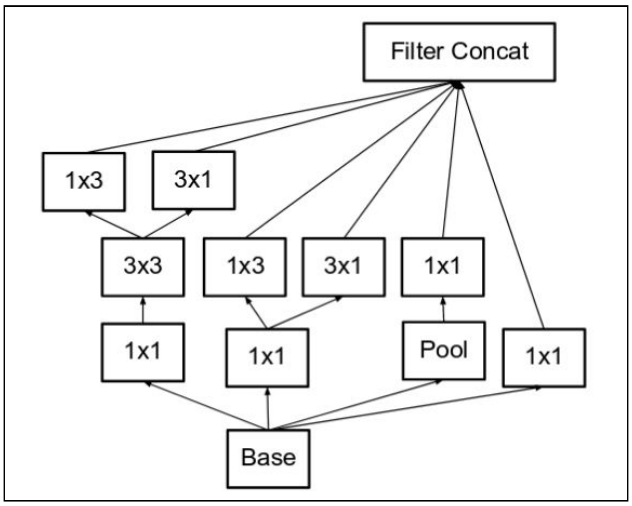


Figure 4.8: Making the inception module wider. This type is equivalent to the module shown in Figure 4.7 (Szegedy et al. 2016).

- The above three principles were used to build three different types of inception modules. The architecture is shown in Table 4.1:

type	patch size/stride or remarks	input size
conv	3×3/2	299×299×3
conv	3×3/1	149×149×32
conv padded	3×3/1	147×147×32
pool	3×3/2	147×147×64
conv	3×3/1	73×73×64
conv	3×3/1	71×71×80
conv	3×3/1	35×35×192
3×Inception	As in figure 5	35×35×288
5×Inception	As in figure 6	17×17×768
2×Inception	As in figure 7	8×8×1280
pool	8 × 8	8 × 8 × 2048
linear	logits	1 × 1 × 2048
softmax	classifier	1 × 1 × 1000

Table 4.1: Here, “figure 5” is Figure 4.6, “figure 6” is Figure 4.7, and “figure 7” is Figure 4.8 (Szegedy et al. 2016).

4.3 Faster R-CNN

R-CNN (Girshick et al., 2014) and Fast R-CNN (Girshick et al., 2015) use selective search to find out the region proposals. Selective search is a slow and time-consuming process affecting the performance of the network. Therefore, Ren et al. (2017) came up with an object detection algorithm that eliminates the selective search algorithm and lets the network learn the region proposals.

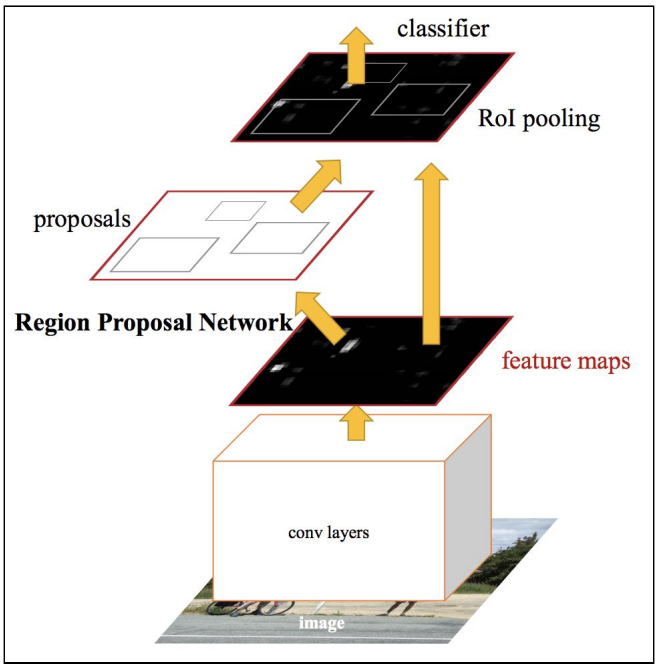


Figure 4.9: Faster R-CNN is a single, unified network for object detection. The RPN module serves as the 'attention' of this unified network (Ren et al., 2017).

Similar to Fast R-CNN, the image is provided as an input to a CNN which provides a convolutional feature map. Instead of using selective search algorithm on the feature map to identify the region proposals, a separate network is used to predict the region proposals. The predicted region proposals are then reshaped using a RoI pooling layer which is then used to classify the image within the proposed region and predict the offset values for the bounding boxes.

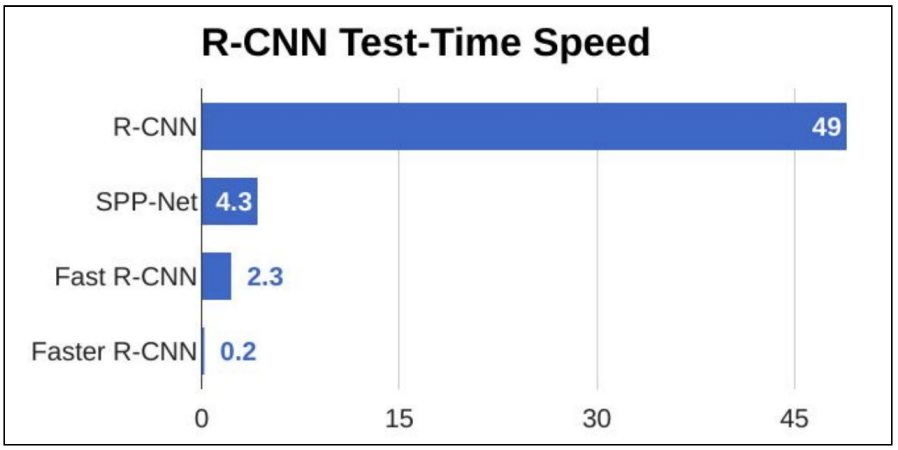


Figure 4.10: Faster R-CNN is much faster than its predecessors. To an extent that it can be used for real-time object detection (John et al., 2019).

4.4 Transfer Learning

Transfer learning is commonly performed on CNNs because of its ability to build accurate models in a timesaving way (Rawat and Wang 2017). With transfer learning, researchers do not need to start the learning process from scratch, but from patterns that have been learned when solving a different problem instead. Previous learnings can hence be leveraged and ultimately, save a lot of time.

Transfer learning is usually expressed through the use of pre-trained models - or in particular, to repurpose them for specific uses (Chollet 2018). A pre-trained model is a model that was trained on a large benchmark dataset to solve a problem. Accordingly, due to the large computational cost of training such models, it is common practice to import and use models from published literature, e.g. ResNet, VGGNet, Mobilenet, etc.

To repurpose a pre-trained model, original classifier is first removed, then a new classifier that fits the new purposes is added, and finally, the model shall be fine-tuned according to one of these three strategies:

1. **Train the entire model.** In this case, the architecture of the pre-trained model is used and trained according to the new dataset. A large dataset and a lot of computational power is normally needed.
2. **Train some layers and leave the others frozen.** Lower layers refer to general features (problem independent), while higher layers refer to specific features (problem dependent). That dichotomy can be followed by choosing the degree of adjustments required on the weights of the network (a frozen layer does not change during training). If the dataset is small and has a large number of parameters, more layers are usually left frozen to avoid overfitting. By contrast, if the dataset is large and the number of parameters is small, the model can be improved by training more layers to the new task since overfitting is not an issue.

3. **Freeze the convolutional base.** This case corresponds to an extreme situation of the train/freeze trade-off. The main idea is to keep the convolutional base in its original form and then use its outputs to feed the classifier. The pre-trained model is used as a fixed feature extraction mechanism, which can be useful if there is a shortage of computational power, the dataset is small, and/or pre-trained model is designed to solve a similar problem.

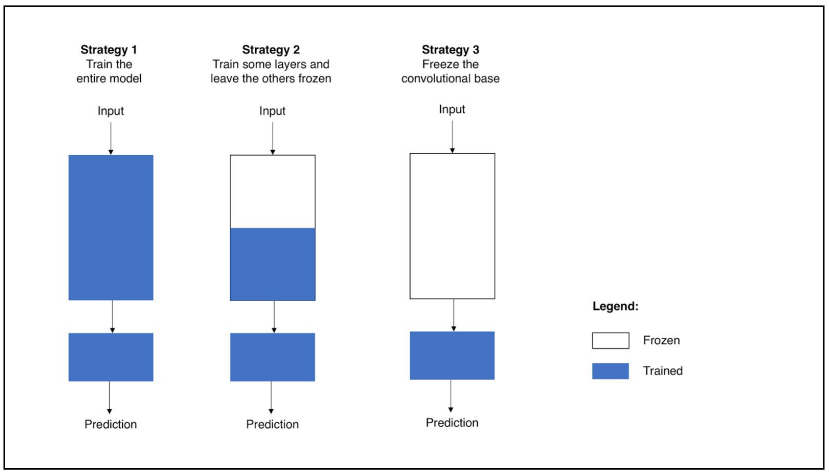


Figure 4.11: Schematics of three fine-tuning strategies (Marcelino, 2018).

Unlike Strategy 3, which application is straightforward, Strategy 1 and Strategy 2 require more handling regarding the learning rate used in the convolutional part. The learning rate is a hyper-parameter that controls how much the weights of the neural network is adjusted. When using a pre-trained model based on CNN, it is smart to use a small learning rate because high learning rates increase the risk of losing previous knowledge. Assuming that the pre-trained model has been well trained, which is a fair assumption, keeping a small learning rate will ensure the CNN weights are not distorted too soon and too much.

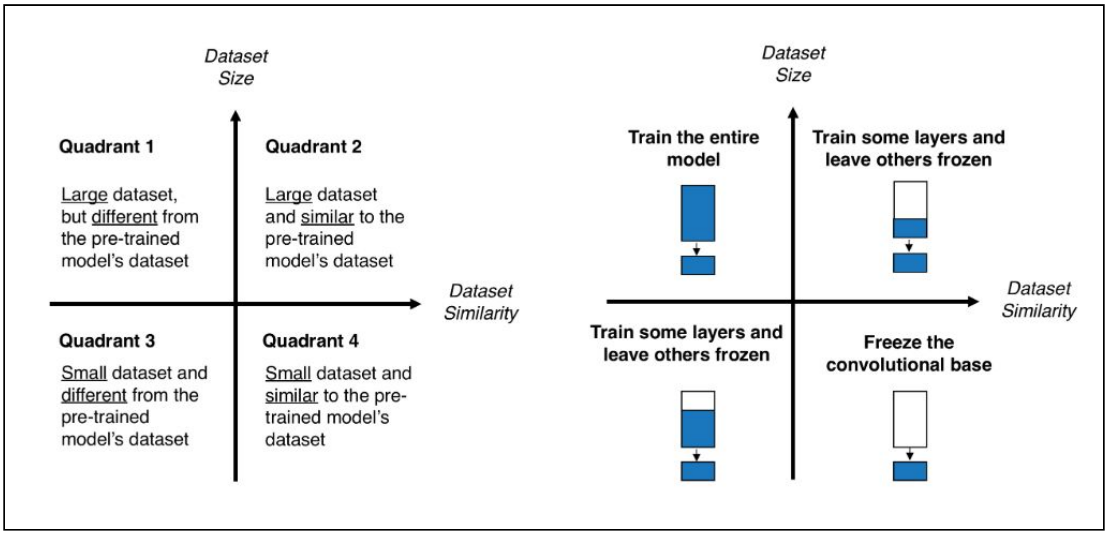


Figure 4.12: Size-Similarity matrix (left) and decision map for fine-tuning pre-trained models (right) (Marcelino, 2018).

4.4 Summary

From this chapter, it is learned that Inception-v1 is an efficient architecture with great accuracy and performance, but Inception-v2 improves upon it to be even better. The researchers (Szegedy et al., 2014) (Szegedy et al., 2016) did it by having 1 x 1 convolutions, global average pooling and the implementation of a new module called Inception module. Inception-v2 made many improvements on the Inception module by optimizing the filter layer and sizes. On the other hand, with transfer learning, previous learnings can be leveraged and save a lot of time for learning. Furthermore, Faster R-CNN is faster than both the original R-CNN and Fast R-CNN, and is the first among the R-CNN architectures that can perform object detection in real-time. And finally, they are generally three strategies in transfer learning, each for different use cases.

Chapter 5

Implementation and Testing

5 Implementation and Testing

A short introduction that describes what will be included in this chapter.

IMPORTANT NOTE TO STUDENTS: Include details about:

- **Implementation**

A detailed description of how you actually carried out the implementation (e.g., coding, etc.) of your system/prototype.

- Include code snippets and descriptions to show how the requirements of the application/prototype have been met –
  - For Smart Campus projects, code snippets of the MQTT protocol for both client side and server side has to be included with explanation.
- Include descriptions of important settings - Setting for server, network protocol, IP, database, security
- Note: this should **not** be a chronological account of the work you carried out.

- **Testing**

All test cases that have been carried out must be provided - To tabulate the test cases in tables

Note: Students may also opt to split Implementation and Testing into 2 separate chapters.

5.1 Sub-section 1 Heading

Sub-sections should be used to divide the chapter into logical parts.

5.1.1 Sub-subsection Heading

Sub-section numbering should be limited to a maximum of 3 levels (e.g. 5.3.1) in order to avoid confusion.

5.2 Sub-section 2 Heading

5.3 Sub-section 1 Heading

Sub-sections should be used to divide the chapter into logical parts.

5.3.1 Sub-subsection Heading

Sub-section numbering should be limited to a maximum of 3 levels (e.g. 5.3.1) in order to avoid confusion.

5.4 Sub-section 1 Heading

Sub-sections should be used to divide the chapter into logical parts.

5.4.1 Sub-subsection Heading

Sub-section numbering should be limited to a maximum of 3 levels (e.g. 5.3.1) in order to avoid confusion.

5.5 Chapter Summary and Evaluation

At the end of each chapter, evaluate the contents stated or discussed in the relevant sub-sections.

Chapter 6 *(if applicable)*

System Deployment

6 System Deployment

This chapter would be applicable for students who have embarked on a real-life industrial project. Students in this case would need to describe how the deployment has been carried out. Some of implementation tasks which need to be described include: training, file conversion or creation, and changeovers.

6.1 System Backup and Risk Management

Describe the procedures to backup the existing system for changeover purpose. Discuss the potential risk(s) for the changeovers and the solution.

6.2 On-site Setup

Describe the preparations to be done prior to the setup of the new system on client's site. Discuss the procedures to setup the new system, schedule, etc.

6.3 Training Procedure

Describe the procedures on the training procedures, contents, schedule, etc.

6.4 Follow-up

Describe the plan or procedures to follow up with the client (company) to verify the system reliability.

6.5 Chapter Summary and Evaluation

At the end of each chapter, evaluate the contents stated or discussed in the relevant sub-sections. For example,

- Problems faced. Describe the various problems faced by students in the course of doing the project.
- Solutions. What have been done to solve the problems?
- What tools and techniques have been used and reasons for using them.

Chapter 7

Discussions and Conclusion

7 Discussions and Conclusion

Each student is required to make an *evaluation of the project* he/she has embarked on. The project evaluation may include the following sections.

IMPORTANT NOTE TO STUDENTS: In this chapter, for problems related to code, hardware, internet connection, etc (where applicable):

- List the technical problems faced and state how they were resolved
- List the unsolved technical problems for future enhancement
- List the achieved objectives/modules
- List the incomplete parts for future enhancement - this is regardless the parts listed in the pre-determined scope. Suggestions for future improvement

7.1 Summary

Summarize the project including the problem and proposed solutions, justification of the choice of tools, techniques and methodologies used in this project.

7.2 Achievements

Students are required to evaluate the project's achievement against project objectives, completion of the project, students' view of the strengths and weaknesses of the work done.

7.3 Contributions

Discuss the creativity, innovativeness, contribution of the proposed system. Explain why the proposed system is necessary. Describe the marketability of the system.

7.4 Limitations and Future Improvements

Identify the limitations of the research or project. Provide suggestions for improvement or further development of the system or research in the future.

7.5 Issues and Solutions

Students are required to describe the various problems faced by students during the project development and explain what has been done to solve the problems. The valuable

experiences gained or lessons learnt through the project as a whole. The issues may include technical issues, project management issues, team dynamics problems, and other difficulties encountered and lessons learnt. How the issues are solved or can be solved to ensure the project can be completed on time or to be improved in the future.

References

Abadi, M. et al., 2016. TensorFlow: A System for Large-Scale Machine Learning. 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16). November 2016 USENIX Association, Savannah, GA, pp. 265–283.

Bottou, L., 2010. Large-Scale Machine Learning with Stochastic Gradient Descent. In: Lechevallier, Y. and Saporta, G., (eds.) Proceedings of COMPSTAT'2010. Physica-Verlag HD, Heidelberg, pp. 177–186.

Bradski, G., 2000. The OpenCV Library. Dr. Dobb's Journal of Software Tools.

Brownlee, J., 2019, A Gentle Introduction to Convolutional Layers for Deep Learning Neural Networks [Online]. Available at: https://machinelearningmastery.com/convolutional-layers-for-deep-learning-neural-networks/ [Accessed: 21 August 2019].

Chakrabarty, A., 2010. An Investigation of Clustering Algorithms and Soft Computing Approaches for Pattern Recognition. Doctoral dissertation, Assam University.

Innovation Enterprise, Big Data Hits the Runway: How Big Data is Changing the Fashion Industry [Online]. Available at: https://channels.theinnovationenterprise.com/articles/8230-big-data-hits-the-runway-how-big-data-is-changing-the-fashion-industry [Accessed: 22 August 2019a].

Chen, H., Gallagher, A. and Girod, B., 2012. Describing clothing by semantic attributes. In: Fitzgibbon, A. et al., (eds.) Computer vision – ECCV 2012. Lecture notes in computer science. Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 609–623.

Chollet, F., 2018. Deep Learning With Python 1st ed., Manning Publications, Shelter Island, New York.

cs231n, Convolutional Neural Networks (CNNs / ConvNets) [Online]. Available at: http://cs231n.github.io/convolutional-networks/ [Accessed: 23 August 2019b].

Ge, Y. et al., 2019. DeepFashion2: A Versatile Benchmark for Detection, Pose Estimation, Segmentation and Re-Identification of Clothing Images. CoRR, abs/1901.07973.

Girshick, R., Donahue, J., Darrell, T. and Malik, J., 2014. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. 2014 IEEE Conference on Computer Vision and Pattern Recognition. June 2014 IEEE.

Girshick, R., 2015. Fast R-CNN. 2015 IEEE International Conference on Computer Vision (ICCV). 7 December 2015 IEEE, pp. 1440–1448.

He, K., Gkioxari, G., Dollar, P. and Girshick, R., 2017. Mask R-CNN. 2017 IEEE International Conference on Computer Vision (ICCV). 22 October 2017 IEEE, pp. 2980–2988.

He, K., Zhang, X., Ren, S. and Sun, J., 2015. Deep Residual Learning for Image Recognition. CoRR, abs/1512.03385.

Hopfield, J.J., 1982. Neural networks and physical systems with emergent collective computational abilities. Proceedings of the National Academy of Sciences of the United States of America, 79(8), pp.2554–2558.

Howard, W.R., 2007. Pattern Recognition and Machine Learning20072Christopher M. Bishop.Pattern Recognition and Machine Learning . Heidelberg, Germany: Springer 2006. i-xx, 740 pp., ISBN: 0-387-31073-8 $74.95 Hardcover. Kybernetes, 36(2), pp.275–275.

Huang, J., Feris, R., Chen, Q. and Yan, S., 2015. Cross-Domain Image Retrieval with a Dual Attribute-Aware Ranking Network. 2015 IEEE International Conference on Computer Vision (ICCV). 7 December 2015 IEEE, pp. 1062–1070.

Jain, A.K., Mao, J. and Mohiuddin, K., 1996. Artificial Neural Networks: A Tutorial. IEEE Computer, 29, pp.31–44.

Jia, Y. et al., 2014. Caffe: Convolutional Architecture for Fast Feature Embedding. arXiv preprint arXiv:1408.5093.

John J. et al., 2019. Review Paper on Object Detection using Deep Learning- Understanding different Algorithms and Models to Design Effective Object Detection Network. International Journal for Research in Applied Science & Engineering Technology (IJRASET), Volume 7 Issue III.

Kepka, J., 1994. The current approaches in pattern recognition. Kybernetika, 30, pp.159–176.

Kiapour, M.H. et al., 2015. Where to buy it: matching street clothing photos in online shops. 2015 IEEE International Conference on Computer Vision (ICCV). 7 December 2015 IEEE, pp. 3343–3351.

Kohavi, R., 2001. A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. International Joint Conference on Arti cial Intelligence (IJCAI), 1995, 14.

Krizhevsky, A., Sutskever, I. and Hinton, G.E., 2012. ImageNet classification with deep convolutional neural networks. Communications of the ACM, 60(6), pp.84–90.

LeCun, Y. et al., 1989. Backpropagation applied to handwritten zip code recognition. Neural Computation, 1(4), pp.541–551.

Lecun, Y., Bottou, L., Bengio, Y. and Haffner, P., 1998. Gradient-based learning applied to document recognition. Proceedings of the IEEE, 86(11), pp.2278–2324.

Lin, M., Chen, Q. and Yan, S., 2013. Network In Network. CoRR, abs/1312.4400.

Lin, T.-Y. et al., 2014. Microsoft COCO: Common Objects in Context. In: Computer Vision – ECCV 2014. Springer International Publishing, pp. 740–755.

Liu, Z. et al., 2016. DeepFashion: Powering Robust Clothes Recognition and Retrieval with Rich Annotations. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 27 June 2016 IEEE, pp. 1096–1104.

Luo, P., Wang, X. and Tang, X., 2013. Pedestrian parsing via deep decompositional network. 2013 IEEE International Conference on Computer Vision. 1 December 2013 IEEE, pp. 2648–2655.

Marcelino P., 2018. Transfer learning from pre-trained models. [Online] Towards Data Science. Available at: https://towardsdatascience.com/transfer-learning-from-pre-trained-models-f2393f124751 [Accessed: 21 August 2019].

McCulloch, W.S. and Pitts, W., 1990. A logical calculus of the ideas immanent in nervous activity. 1943. Bulletin of Mathematical Biology, 52(1–2), p.99–115; discussion 73.

Nandhakumar, N. and Aggarwal, J.K., 1985. The artificial intelligence approach to pattern recognition—a perspective and an overview. Pattern recognition, 18(6), pp.383–389.

Nayak, S., 2018, Understanding AlexNet [Online]. Available at: https://www.learnopencv.com/understanding-alexnet/ [Accessed: 21 August 2019].

Nicholson, C., A Beginner's Guide to Neural Networks and Deep Learning [Online]. Available at: https://skymind.ai/wiki/neural-network.html [Accessed: 22 August 2019].

Nielsen, M., 2019, How the backpropagation algorithm works [Online]. Available at: http://neuralnetworksanddeeplearning.com/chap2.html [Accessed: 21 August 2019].

Nievergelt, J., 1969. R69-13 Perceptrons: An Introduction to Computational Geometry. IEEE Transactions on Computers, C-18(6), pp.572–572.

Programmable Web, Deepomatic Fashion Apparel Detection API [Online]. Available at: https://www.programmableweb.com/api/deepomatic-fashion-apparel-detection [Accessed: 21 August 2019c].

Quaknine A. 2018. Review of Deep Learning Algorithms for Object Detection. [Online] Medium. Available at: https://medium.com/zylapp/review-of-deep-learning-algorithms-for-object-detection-c1f3d437b852 [Accessed: 21 August 2019].

Rawat, W. and Wang, Z., 2017. Deep convolutional neural networks for image classification: A comprehensive review. Neural Computation, 29(9), pp.2352–2449.

Redmon, J., Divvala, S., Girshick, R. and Farhadi, A., 2016. You Only Look Once: Unified, Real-Time Object Detection. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). June 2016 IEEE.

Ren, S., He, K., Girshick, R. and Sun, J., 2017. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. IEEE transactions on pattern analysis and machine intelligence, 39(6), pp.1137–1149.

Russakovsky, O. et al., 2015. ImageNet large scale visual recognition challenge. International journal of computer vision, 115(3), pp.211–252.

Schalkoff, R.J., 2007. Pattern Recognition. In: Wah, B.W., (ed.) Wiley encyclopedia of computer science and engineering. John Wiley & Sons, Inc., Hoboken, NJ, USA.

Schmidhuber, J., 2015. Deep learning in neural networks: an overview. Neural Networks, 61, pp.85–117.

Shorten, C., 2019, Introduction to ResNets [Online]. Available at: https://towardsdatascience.com/introduction-to-resnets-c0a830a288a4 [Accessed: 21 August 2019].

Simonyan, K. and Zisserman, A., 2015. Very Deep Convolutional Networks for Large-Scale Image Recognition. 2015 arXiv.

Skalski, P., 2018, Preventing Deep Neural Network from Overfitting [Online]. Available at: https://towardsdatascience.com/preventing-deep-neural-network-from-overfitting-953458db800a [Accessed: 21 August 2019].

Szegedy, C. et al., 2014. Going Deeper with Convolutions. CoRR, abs/1409.4842.

Szegedy, C. et al., 2015. Rethinking the Faster R-CNN for Computer Vision. CoRR, abs/1512.00567.

Szegedy, C. et al., 2016. Rethinking the Inception Architecture for Computer Vision. 2016 IEEE

Conference on Computer Vision and Pattern Recognition (CVPR). June 2016 IEEE.

TensorFlow (2019). ObjectDetection API not suitable for tf 2.0.0-alpha0. [online] GitHub. Available at: https://github.com/tensorflow/models/issues/6423 [Accessed: 21 August 2019].

TensorFlow (2019). Tensorflow detection model zoo. [online] GitHub. Available at: https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md [Accessed: 21 August 2019].

Tsang, S.-H., 2018a, Review: GoogLeNet (Inception v1)— Winner of ILSVRC 2014 (Image Classification) [Online]. Available at: https://medium.com/coinmonks/paper-review-of-googlenet-inception-v1-winner-of-ilsvlc-2014-image-classification-c2b3565a64e7 [Accessed: 22 August 2019].

Tsang, S.-H., 2018b, Review: VGGNet — 1st Runner-Up (Image Classification), Winner (Localization) in ILSVRC 2014 [Online]. Available at: https://medium.com/coinmonks/paper-review-of-vggnet-1st-runner-up-of-ilsvlc-2014-image-classification-d02355543a11 [Accessed: 22 August 2019].

Werbos, P., 1974. Beyond regression : new tools for prediction and analysis in the behavioral sciences /. Undergraduate thesis,

Wired, 2019, Here's how Nike, Alibaba and Walmart are reinventing retail [Online]. Available at: https://www.wired.co.uk/article/future-of-retail [Accessed: 22 August 2019].

Yang, M. and Yu, K., 2011. Real-time clothing recognition in surveillance videos. 2011 18th IEEE International Conference on Image Processing. 11 September 2011 IEEE, pp. 2937–2940.

Yiu, T., 2019, Understanding Neural Networks [Online]. Available at: https://towardsdatascience.com/understanding-neural-networks-19020b758230 [Accessed: 22 August 2019].

Zadeh, L.A., 1965. Fuzzy sets. Information and Control, 8(3), pp.338–353.

Zhou, X., 2018. Understanding the Convolutional Neural Networks with Gradient Descent and Backpropagation. Journal of Physics: Conference Series, 1004, p.012028.

Appendices

In order to enhance better understanding of the project, students should as far as possible include all directly relevant materials, figures or diagrams in the **main body** rather than in the Appendix. The appendix is reserved only for items which may not directly be relevant or essential to enhance a reader's understanding of the project, or which may interrupt the smooth reading of the project document (for example being too voluminous).

Appendices should only include supportive materials **directly referred** to in the writing and should be kept to a **minimum**, e.g. selected pages of an annual report, not the entire document. Examples of items included in Appendices are:

- Company's report and documentation, such as sample invoice, purchase order form, etc.
- Project meeting documentation e.g. minutes of meetings, tracking documents, memos etc.
- Questionnaires and results, interview questions and results, pilot test and results, observation sheet and results, experiment test plan and results, etc.
- Analysis/design diagrams (only those not incorporated in the main body of the report).

If there is more than one appendix, they should be identified as A, B, etc (e.g. Appendix A). Formulae and equations in appendices should be given separate numbering: Eq. (A.1), Eq. (A.2), etc.; in a subsequent appendix, Eq. (B.1) and so on. Similarly for tables and figures: Table A.1; Fig. A.1, etc.

IMPORTANT NOTE TO STUDENTS

APPENDIX *n* User Guide

- List the username and password of multiple roles (if applicable)
- Provide clear screen shots of each page, explain the functions of each button

As a rough guide, the user guide should include the following sections:

System Document

In this section, students should provide the following pieces of information:

- **System (hardware and software requirements)**. Students should describe the minimum hardware and software requirements to install the software application which has been developed by the students, for example, DBMS, OS, program development tools etc.
- **Installation**. Under this section, students should create an 'installation' CD and provide a brief step-by-step guide on how a new user can install the software on a computer system. Students should indicate any special setup information, such as the specific location of placing the database files, the SQL statements to add tables,

etc. Software source code should also be included in this CD. Students are not required to print out the software code.

Operation Document

Under this section, students are required to provide a brief step-by-step guide on how to use the installed software. The guides should teach the user how to run the software and use its major functions and features. For example, steps show guide the users on how to run the system, e.g. to use an executable file or to use the IDE. The login information such as username and password for each of the different users (or roles) must be provided. Some screen interfaces would be useful.

APPENDIX *n+1* Developer Guide*

*To be included for ***Smart Campus Projects*** and ***Real-Life Projects***. This section should include the following:

- List the necessary software, installer, API, library that must be installed
- List the authentication details, such as username and password for all security

Other Appendices:

- Supporting documents
- Non-disclosure agreement (if applicable)
- Other detailed documentation, such as important references, interview results, survey results, etc.

This page is intentionally left blank to indicate the back cover. Ensure that the back cover is black in color.